

Trinity Release 3.12.X

Command Line Reference Guide

Sales Office: +1 (301) 975-1000
Technical Support: +1 (301) 975-1007
E-mail: support@patton.com
WWW: www.patton.com

Patton Electronics Company, Inc.

7622 Rickenbacker Drive, Gaithersburg, MD 20879 USA

Tel: +1 (301) 975-1000 • **Fax:** +1 (301) 869-9293 • **Support:** +1 (301) 975-1007

Web: www.patton.com • **E-mail:** support@patton.com

Copyright Statement

Copyright © 2009–2017, Patton Electronics Company. All rights reserved.

Trademark Statement

The terms *Trinity* and *ForeFront* are trademarks of Patton Electronics Company. All other trademarks presented in this document are the property of their respective owners.

Notices

The information contained in this document is not designed or intended for use as critical components in human life-support systems, equipment used in hazardous environments, or nuclear control systems. Patton Electronics Company disclaims any express or implied warranty of fitness for such uses.

The information in this document is subject to change without notice. Patton Electronics assumes no liability for errors that may appear in this document.

Any software described in this document is furnished under license and may be used or copied only in accordance with the terms of such license.

Summary Table of Contents

1	System Overview	51
2	Configuration Concepts	57
3	Command Line Interface (CLI)	62
4	Accessing the CLI	67
5	Creating CLI Action Scripts	79
6	System Image Handling	84
7	Configuration File Handling	88
8	System Licensing	98
9	AAA Configuration	126
10	Basic System Management	135
11	Programmable System-Event Configuration	149
12	Alarm Management	182
13	Auto Provisioning of Firmware and Configuration	185
14	Ethernet Port Configuration	197
15	Configuring a Cellular Modem	205
16	Hardware Switching	213
17	DSL Port Configuration	248
18	Context Bridge	260
19	Spanning Tree Configuration	264
20	PPP Configuration	270
21	IP Context Overview	280
22	IP Interface Configuration	289
23	IP Routing	300
24	Internet Protocol Security (IPsec) Configuration	314
25	OpenVPN Configuration	329
26	Generic Routing Encapsulation (GRE) Configuration	340
27	Border Gateway Protocol (BGP) Configuration	345
28	Load Balancing and Failover Configuration	357
29	Fast-Path	369
30	NAT/NAPT Configuration	372
31	DHCP Configuration	380

32	DNS Configuration	386
33	SNTP Client Configuration	390
34	SNMP Configuration	393
35	Public-Key Infrastructure (PKI)	408
36	Quality of Service (QoS) Overview	423
37	Profile Service-Policy Configuration	427
38	Access Control List Configuration	440
39	Classifier Configuration	450
40	Packet Matching	458
41	SIP Profile Configuration	464
42	VoIP Profile Configuration	470
43	PSTN Profile Configuration	495
44	CS Context Overview	499
45	CS Interface Configuration	521
46	Tone Configuration	530
47	Authentication Service	537
48	Location Service	540
49	Call Router Configuration	561
50	Global SIP Configuration	634
51	SIP Overload Configuration	638
52	SIP Interface Configuration	647
53	SIP Survivability	665
54	TLS Configuration for SIP	682
55	TLS Configuration for Lync	708
56	Secure SIP Applications	725
57	Secure Real-Time Protocol (SRTP) Configuration	734
58	SIP Security	740
59	SIP Call-router Services	771
60	Context SIP Gateway Overview	777
61	ISDN Overview	793
62	ISDN Configuration	798
63	ISDN Interface Configuration	806
64	FXS Interface Configuration	823
65	PRI Port Configuration	872

66	E1/T1 Port Configuration	878
67	BRI Port Configuration	880
68	FXS Port Configuration	885
69	Debug and Monitoring	906
70	Contacting Patton for Assistance	920
A	Trinity Architecture Terms and Definitions	923
B	Command Summary	929
C	Glossary of Terms	932

Table of Contents

Summary Table of Contents	3
Table of Contents	6
List of Figures	39
List of Tables	42
About this guide	43
Audience.....	43
How to read this guide	43
Structure.....	43
Precautions	48
Typographical conventions used in this document.....	48
General conventions	48
Service and support	49
Patton support headquarters in the USA	49
Alternate Patton support for Europe, Middle East, and Africa (EMEA)	49
Warranty Service and Returned Merchandise Authorizations (RMAs).....	49
Warranty coverage	49
Returns for credit	49
Return for credit policy	50
RMA numbers	50
Shipping instructions	50
1 System Overview	51
Introduction.....	52
Trinity embedded software.....	53
Applications	53
Carrier networks	54
Enterprise networks	54
LAN telephony	55
2 Configuration Concepts	57
Introduction.....	58
Contexts and Gateways.....	59
Context	59
Gateway	59
Interfaces, Ports, and Bindings	60
Interfaces	60
Ports and circuits	60
Bindings	60
Profiles and Use commands.....	61
Profiles	61
Use Commands	61

3	Command Line Interface (CLI)	62
	Introduction	63
	Command modes	63
	CLI prompt	63
	Navigating the CLI	63
	Initial mode	63
	System changes	64
	Configuration	64
	Changing Modes	64
	Command editing	64
	Command help	64
	The No Form	64
	Command completion	64
	Command history	65
	Command Editing Shortcuts	65
	Timed Execution of CLI Command	65
4	Accessing the CLI	67
	Introduction	68
	Accessing the Trinity CLI task list	68
	Accessing via the console port	69
	Console port procedure	69
	Accessing via a secure configuration session over SSH	69
	Accessing via a Telnet session	70
	Telnet Procedure	70
	Using an alternate TCP listening port for the Telnet or SSH server	70
	Disabling the Telnet or SSH server	70
	Logging on	70
	Selecting a secure password	71
	Password encryption	72
	Factory preset superuser account	72
	Configuring operators, administrators, and superusers	72
	Creating an operator account	72
	Creating an administrator account	73
	Creating a superuser account	74
	Displaying the CLI version	75
	Displaying account information	75
	Checking identity and connected users	75
	Command index numbers	76
	Ending a Telnet, SSH or console port session	78
5	Creating CLI Action Scripts	79
	Introduction	80
	Action Script Task List	80
	Creating an Action Script	80

Conditions	80
Context CS Events	81
SIP Interface Events	81
Context IP Events	81
SIP Gateway Events	82
System NTP Events	82
System Timer Events	82
Actions	83
6 System Image Handling	84
Introduction	85
System image handling task list	85
Displaying system image information	85
Displaying Update Status Information	86
Copying system images from a network server to flash memory	86
Switch to the inactive image	87
Erase inactive image on dual-image system	87
7 Configuration File Handling	88
Introduction	89
Understanding Configuration Files	89
Shipping Configuration	90
Configuration File Handling Task List	90
Copying Configurations Within the Local Memory	91
Replacing the Startup Configuration with a Configuration from Flash Memory	92
Copying Configurations To and From a Remote Storage Location	93
Replacing the Startup Configuration with a Configuration Downloaded from TFTP Server	93
Displaying Configuration File Information	94
Modifying the Running Configuration at the CLI	95
Modifying the Running Configuration Offline	96
Deleting a Specified Configuration	97
8 System Licensing	98
Introduction	99
Install licenses permanently	101
Installing licenses	101
Inspecting installed licenses	102
Erasing installed licenses	105
Configure a Patton device as License Server	105
Prepare the system	107
Configure the NodeMS-Server	108
Create organizations for tenants	108
Create sub-organizations within an organization	110
Define which and how many licenses a particular device is able to lease	112
Create a nodems-device profile	113
Link devices to a nodems-device profile	115

Inspect the license-lease database	116
Configure Patton devices to lease licenses from a License Server	121
Prepare the system	121
Connect a Patton device to a License Server	122
Display information about the License Server connection	124
9 AAA Configuration	126
Introduction	127
The AAA Component	127
General AAA Configuration	128
Configuring TACACS+ client	129
Configuring TACACS+ server	129
Authentication	129
Authorization	129
Server configuration example	130
Configuring RADIUS clients	132
Configuring RADIUS server	132
Attributes in the RADIUS Authentication request message	132
Attributes in RADIUS Authentication request for Authorization	132
Attributes in RADIUS Accept message	133
10 Basic System Management	135
Introduction	136
Basic System Management Configuration Task List	136
Managing Feature License Keys	137
Showing System Resources	138
Setting System Parameters	138
Setting the System Banner	140
Setting Time and Date	140
Configuring Daylight Savings Time Rules	141
Display Clock Information	142
Display Time Since Last Restart	142
Configuring and starting the web server	142
Restarting the system	143
Displaying the System Logs	143
Displaying the System Logs	144
Exporting System Logs and Reports	145
Configuring the blink interval	146
Configuring the Syslog Client	146
Factory Reset	147
Reset Button	147
Trinity Performance Tracker	147
11 Programmable System-Event Configuration	149
Introduction	151
System variables	151

User-Defined expressions	152
Actions	153
Expression configuration task list.....	153
Collect information about the variables that build the expression	154
Display existing system variables and expressions	154
Tracking real-time changes of system variables and expressions	155
Validate an expression (on-the-fly computation).....	155
Create/Modify an expression	156
Extend a system variable by an expression.....	157
Create/Modify an expression family.....	158
Wildcards and regular expressions in context-family names	159
Delete an expression	161
Expression Syntax.....	162
Data Types	162
Booleans	162
Numbers	162
Time Stamps	163
Text Strings	163
Errors	163
Operators	164
Logical Operators	164
Bitwise Operators	165
Arithmetic Operators	166
Comparison Operators	166
Operator Precedence	167
Functions	168
Logical Functions	169
Bitwise Functions	170
Arithmetic Functions	170
Comparison Functions	172
Set Functions	173
Time/Date Functions	173
Temporal Functions	174
Example Expressions	177
State Profiles.....	177
Default OVERLOAD state profile	179
Check the configuration of state profiles	180
Debug state transitions	181
12 Alarm Management	182
Introduction.....	183
Alarm Configuration Task List.....	183
Viewing alarms	183
Changing alarm options	184

13	Auto Provisioning of Firmware and Configuration	185
	Introduction	186
	Provisioning Profile	186
	Creation	186
	Destination	186
	Destination Script	186
	Destination Configuration	187
	Destination Upload	187
	Locations	188
	Placeholders in Locations	188
	Conditional Placeholders in Locations	189
	Activation	190
	Include Command	190
	User authentication	191
	Server authentication	191
	Client Authentication	191
	Private Key	191
	Own Certificate Chain	191
	Disable Client Authentication	192
	TLS Profile commands used from provisioning	192
	TR-069	194
	PKI commands used from provisioning	195
	Using Provisioning	196
	Provisioning Reset	196
	Provisioning Status	196
14	Ethernet Port Configuration	197
	Introduction	198
	Ethernet Port Configuration Task List	198
	Entering the Ethernet Port Configuration Mode	198
	Configuring Medium for an Ethernet Port	198
	Binding an Ethernet Port	199
	Multiple IP Addresses on Ethernet Ports	200
	Configuring a VLAN	201
	Configuring Layer-2-CoS to Service-class Mapping for a VLAN	202
	Closing an Ethernet Port	203
15	Configuring a Cellular Modem	205
	Introduction	206
	Supported Cellular Modems	206
	Virtual Ports vs. Physical Modems	206
	Cellular Modem Configuration Task List	207
	Creating a Virtual Port and Linking It to a Physical Modem	207
	Configuring the Cellular Network Parameters	208
	Configuring the IP Address	208

Enabling the Cellular Modem	208
Displaying the Cellular Modem's Configuration and Status	209
Debugging the Cellular Modem	210
Troubleshooting.....	210
My cellular modem is not getting an IP address	210
My cellular modem has a static public IP address but I can't access my device using it	211
16 Hardware Switching	213
Introduction	215
Switch Groups.....	215
Switch Group Configuration Task List	216
Create a switch group	216
Bind switch group to an IP interface	216
Create switch group interfaces	216
Bind ports to switch group interface	217
Examples	217
LAN/WAN Configuration	217
Configuring Two LANs	218
VLAN (802.1p/Q)	218
VLAN configuration task list	219
Configure switch mode	219
Enter switch group interface configuration mode	219
Permit untagged packets	219
Permit tagged packets	220
Encapsulate untagged traffic	220
Encapsulate all traffic	221
Examples	221
Example 1: Interface Isolation	221
Example 2: VLAN Tagging	222
Example 3: Q-in-Q	224
Access Control List Configuration.....	224
About Access Control Lists (ACLs)	224
What ACLs Do	224
Why You Should Configure ACLs	225
Features of Access Control Lists	225
Access Control List (ACL) Configuration Task List	226
Mapping the Goals of the ACL	226
Creating an ACL Profile and Entering Configuration Mode	226
Adding and Deleting a Filter Rule to the Current ACL Profile	227
Binding and Unbinding an ACL Profile to a Switch Port	230
Displaying an ACL Profile	230
QoS Traffic Scheduler.....	231
About QoS	231
Packet walkthrough	231

QoS Traffic Scheduler Configuration Task List	232
Create a class of service profile and assign its traffic class	232
Create an access control list profile and create classifier rules	233
Binding an access control list to the receiving switch port	233
Configure traffic classes' scheduling modes	233
Binding a service policy to the transmitting switch port	234
Configuring transmit rate shaping for a switch port	235
Example	236
ToS Stripping and Prioritization.....	237
About ToS stripping and prioritization	237
ToS Stripping and Prioritization Configuration Task List	237
Create a class of service profile and configure its VLAN priority and/or DSCP	238
Create an access control list profile and create classifier rules	238
Binding an access control list to the receiving switch port	238
Example	239
MAC Filter Configuration.....	239
Ethernet Switch MAC Filter Configuration Task List	239
Creating a MAC Filter Profile and Enter Configuration Mode	240
Adding a Filter Rule to the Current MAC Filter Profile	240
Binding and Unbinding a MAC Filter to an Ethernet Switch Port	240
Displaying a MAC Filter Profile	241
Trunk Configuration.....	242
Ethernet Switch Trunk Configuration Task List	242
Creating a Trunk Profile and Enter Configuration Mode	242
Binding and Unbinding a Trunk Profile to an Ethernet Switch Port	242
Displaying an Ethernet Switch Trunk	243
Debugging an Ethernet Switch Trunk	243
Ethernet Service Policy Configuration	244
About QoS	244
Packet Walkthrough	244
Ethernet Switch Service Policy Configuration Task List	246
Configure Traffic Class for Priority Scheduling	246
Configure Traffic Class for Shared Scheduling	246
Binding a Service Policy to an Ethernet Switch Port	246
17 DSL Port Configuration	248
Introduction.....	249
G.SHDSL EFM Setup	249
Configuring the Mode for the G.SHDSL Connection	249
Configuring the Annex Type for the G.SHDSL Connection	250
Configuring the Payload Data Rate for the G.SHDSL Connection	250
Configuring the TCPAM for the G.SHDSL connection	251
DSL Emergency Freeze	251
DSL TX Power Increase	251

DSL Suspect Mode	252
Multiport G.SHDSL Devices	252
Configuring the Profile for the G.SHDSL Connection	252
VDSL/ADSL Configuration.....	253
Introduction	253
Configuration Task List	253
Configure Physical Parameters	254
Configure PTM Protocol Stack	254
IPoE	254
PPPoE	255
Create ATM PVC	256
Configure ATM Protocol Stack	256
IPoEoA	257
PPPoEoA	257
Troubleshooting DSL Connections.....	258
Checking DSL connection status with show command	258
Link State	259
Debugging the DSL connection with the DSL debug command.	259
18 Context Bridge	260
Introduction	261
Bridge group Configuration Task List	261
19 Spanning Tree Configuration	264
Introduction	265
Spanning Tree Configuration Task List.....	265
Configuring Global Spanning Tree Parameters	266
Configuring Per-tree Spanning Tree Parameters	266
Configuring Per-port/Per-tree Spanning Tree Parameters	267
Enabling Spanning Tree on a Port	267
Debugging Spanning Tree	267
Spanning Tree Configuration Example	268
20 PPP Configuration	270
Introduction.....	271
PPP Configuration Task List.....	272
Creating an IP Interface for PPP	272
Creating a PPP Session	273
Configuring a PPPoE Session	275
Creating a PPP Profile	275
Displaying PPP Configuration Information	277
Debugging PPP	277
Sample Configurations	278
PPP Over Ethernet (PPPoE)	278
Without authentication, encapsulation multi, with NAPT	278
With authentication, encapsulation PPPoE	279

21	IP Context Overview	280
	Introduction	281
	Packet Processing in the IP Context	282
	Classifier	284
	Network Address Port Translation (NAPT)	284
	Routing-table Selection	284
	Access Control Lists (ACL)	284
	Routing	284
	Packet Processing To/From Local Applications	285
	IP Context Overview Configuration Task List.....	285
	Planning Your IP Configuration.....	286
	IP Interface Related Information	286
	QoS Related Information	286
	Configuring Physical Ports	286
	Creating and Configuring IP Interfaces	287
	Configuring Packet Classification	287
	Configuring Network Address Port Translation (NAPT)	287
	Configuring Static IP Routing	287
	Configuring Access Control Lists (ACL)	288
	Configuring Quality of Service (QoS)	288
22	IP Interface Configuration	289
	Introduction	290
	IP Interface Configuration Task List	290
	Creating an IP Interface	290
	Deleting an IP Interface	291
	Setting the Static IP Address and Network Mask	292
	Deleting an IP Address	293
	Displaying IP Interface Information	294
	Displaying Dynamic ARP Entries	296
	ARP Garbage-Collector	296
	Gratuitous ARP	296
	Testing Connections with the Ping Command	297
	Traceroute Command	297
	Debugging the IP Configuration	298
23	IP Routing	300
	Introduction	301
	Basic Routing	301
	Static Routes	301
	Configuring static routes	301
	System Routes	302
	Dynamic Routes	302
	Show Routes	303
	Basic Static Routing Example	303

Policy Routing.....	304
Routing Tables	305
Creating a table	306
Configuring static routes	306
Show routes	306
Traffic Assignment	306
Assign an IP Interface	307
Assignment by Rules	308
Assignment by Traffic-Class	310
24 Internet Protocol Security (IPsec) Configuration	314
Introduction	316
About IPsec	316
IKE	316
Data transport	316
IPsec configuration task list	317
Install License	317
Configure Time.....	317
Configure Public-Key Infrastructure (PKI)	317
Configure IPsec profile	318
Creating an IPsec profile and entering configuration mode	319
Entering the Phase 1 configuration mode	319
Configuring the Phase 1	319
Entering the Phase 2 configuration mode	319
Configuring the Phase 2	320
Configure IPsec tunnel	320
Local	320
Remote	320
Local identity	320
Remote identity	320
Bind	320
IPsec profile	320
Authentication method	321
Pre-shared key	321
Public key	321
Match	321
Creating an IPsec tunnel and entering configuration mode	322
Configuring the IPsec tunnel	322
Entering the authentication configuration mode	322
Configuring the authentication	323
Configure traffic selectors	323
Entering configuration mode	324
Configuring the traffic selectors	324
Configure road warrior vs site-to-site VPN	324

Examples	325
IPsec profile configuration	325
IP interface configuration	326
IPsec tunnel configuration	326
Troubleshooting.....	327
Diffie-Hellman groups and encryption/integrity algorithms	327
Diffie-Hellman groups	327
Encryption algorithms	327
Integrity algorithms	327
25 OpenVPN Configuration	329
Introduction	330
OpenVPN Configuration Task List (Quick Start).....	331
Configuring NTP	331
Configuring OpenVPN Tunnel	331
OpenVPN Configuration Task List (Full Reference).....	332
Configuring NTP¶	332
Creating a Public-Key Infrastructure (PKI)¶	332
Creating a Certificate Authority (CA) (Server Only)	333
Obtaining the Keys and Certificates (Server)	333
Obtaining the Keys and Certificates (Client)	333
Configuring the TLS Profile	334
Configuring OpenVPN Tunnel	335
Debugging OpenVPN.....	336
Sample Configurations	338
Common Configuration	338
Application 1: Point-to-point	339
Application 2: Point-to-multipoint	339
26 Generic Routing Encapsulation (GRE) Configuration	340
Introduction	341
GRE Configuration Task List.....	341
Configuring GRE Tunnel	341
Debugging GRE Tunnel	342
Sample Configurations	343
Common Configuration	343
27 Border Gateway Protocol (BGP) Configuration	345
Introduction	346
About Border Gateway Protocol (BGP).....	346
BGP configuration task list.....	346
Starting BGP	346
Configure BGP general behavior	347
Always compare MED	347
Deterministic MED	347
Log neighbor changes	348

Graceful restart	348
Timers	348
Distance	349
Redistribute	349
Configure BGP Networks	349
Configure BGP Neighbors	350
Configure BGP neighbor behavior	350
Configure BGP neighbor route-mappings	351
Configure route-map profile	351
Match rules	352
Set rules	353
Examples	354
Troubleshooting	355
28 Load Balancing and Failover Configuration	357
Introduction	358
Policy Routing	359
Heartbeat Profile	359
Load Balancing Service	359
Load Balancing and Failover Configuration Task List	359
Create a Heartbeat Profile (Optional)	360
Start a Heartbeat (Optional)	361
Configure the Routing Table(s) for the DHCP or Cell-modem Default Route (Optional)	361
Create a Load Balancer Service	361
Assign Traffic to a Load Balancer Service	363
Display the Heartbeat Status (Optional)	364
Display the Load Balancer Service Status (Optional)	364
Examples	365
Load Balancing and Failover Example	365
Troubleshooting	367
I lose management connectivity to my device when all my WAN links are down	367
After all my WAN links go down, they never come back up	367
29 Fast-Path	369
Introduction	370
Fast-Path Configuration	370
30 NAT/NAPT Configuration	372
Introduction	373
Dynamic NAPT	373
Static NAPT	374
Dynamic NAT	375
Static NAT	375
NAPT traversal	376
NAT/NAPT Configuration Task List	376
Creating a NAPT Profile	376

Configuring a NAPT DMZ host	377
Activate NAT/NAPT	378
Displaying NAT/NAPT Configuration Information	378
31 DHCP Configuration	380
Introduction	381
DHCP-client Configuration Tasks	382
Configure an IP interface for DHCP	382
Release or Renew a DHCP Lease Manually (advanced)	383
Remove a DHCP address from an IP interface	383
Capture Debug Output from DHCP-client	383
32 DNS Configuration	386
Introduction	387
DNS Configuration Task List	387
Enabling the DNS Resolver	387
Enabling the DNS Relay	388
33 SNTP Client Configuration	390
Introduction	391
NTP Client Configuration Task List	391
Enabling/Disabling the NTP Management Component	391
Enabling NTP Options	391
Examples	392
Run the following to see the NTP configuration settings	392
Run the following to see the NTP status	392
34 SNMP Configuration	393
Introduction	394
Simple Network Management Protocol (SNMP)	394
SNMP Basic Components	394
SNMP Basic Commands	394
SNMP Management Information Base (MIB)	395
Network Management Framework	395
Identification of a Patton device via SNMP	396
SNMP Tools	396
SNMP Configuration Task List	396
Setting Basic System Information	396
Setting Access Community Information	399
Setting Allowed Host Information	400
Authentication and Encryption	400
Specifying the Default SNMP Trap Target	401
Displaying SNMP Related Information	401
Using the ManageEngine SNMP Utilities	402
Using the MibBrowser	402
Using the TrapViewer	403

Standard SNMP Version 1 Traps	406
SNMP Interface Traps	407
35 Public-Key Infrastructure (PKI)	408
Introduction	409
Overview	409
Architecture	409
Symmetric encryption and the key-distribution problem	410
Asymmetric encryption	411
CA-signed certificate enrollment	412
Self-signed certificate enrollment	413
PKI-related commands	414
Private-key handling	414
Public-key handling	415
Certificate-request handling	415
Own-certificate handling	418
Trusted-certificate handling	420
Generated default files	422
36 Quality of Service (QoS) Overview	423
Introduction	424
Packet Classification	424
Type-of-Service (TOS)/Class-of-Service (CoS) Mapping	425
37 Profile Service-Policy Configuration	427
Introduction	428
Applying Scheduling at the Bottleneck	428
Using Traffic Classes	428
Patton DownStreamQoS™	428
Introduction to Scheduling	429
Priority	429
Weighted fair queuing (WFQ)	429
Shaping	429
Handling of bursts	429
Hierarchy	430
Quick References	430
Setting the Modem Rate	430
Configure DownStreamQoS™	430
Configuration Example: Common application case	431
Service-Policy configuration task list	432
Creating a service-policy profile	432
Configure Link arbiter	433
Rate limit	433
Arbiter mode	433
Source traffic-class	434
Hierarchical profile service-policy	434

	Share for weighted-fair-queuing	434
	Bit-rate for shaper	435
	Assigning absolute priority	435
	Real time traffic	435
	Queue length	435
	Queue type	436
	Discarding excess load	436
	Set QoS-related IP header field	436
	Binding a classifier profile to the outbound traffic of an IP interface	437
	Troubleshooting.....	438
38	Access Control List Configuration	440
	Introduction.....	441
	About Access Control Lists (ACLs).....	441
	What Access Lists Do	441
	Why You Should Configure Access Lists	442
	When to Configure Access Lists	442
	Features of Access Control Lists	443
	Access Control List Configuration Task List.....	443
	Mapping Out the Goals of the Access Control List	443
	Creating an Access Control List Profile and Enter Configuration Mode	444
	Adding and Deleting a Filter Rule to the Current Access Control List Profile	444
	Binding and Unbinding an Access Control List Profile to an IP Interface	445
	Displaying an Access Control List Profile	447
	Examples	448
	Denying a Specific Subnet	448
	Denying Traffic Between Two Interfaces	449
	Permit Only Traffic Generated From LAN	449
39	Classifier Configuration	450
	Introduction.....	451
	About the Classifier	451
	What the Classifier Does	451
	How the Classifier Works	451
	Classifier Configuration Task List	452
	Mapping the Goals of the Classifier	452
	Creating a Classifier Profile and Enter Configuration Mode	452
	Adding a Rule to the Current Classifier Profile	453
	Binding and Unbinding a Classifier Profile To/From an IP Interface to Tag Incoming/Outgoing Packets	453
	Binding and Unbinding a Classifier Profile to Tag Locally-generated Packets	455
	Displaying a Classifier Profile	456
40	Packet Matching	458
	Introduction.....	459
	Criteria.....	459
	Connection State	459

Traffic Class	459
Source MAC Address	459
Ethernet Packet Type	459
ToS	459
Precedence	460
DSCP	460
ECN	460
Length	460
TTL	460
Protocol	460
Source IP Address	460
Destination IP Address	460
ICMP Type/Code	460
Source Port	460
Destination Port	460
TCP Flags	460
TCP Option	460
TCP MSS	460
Command Line Syntax.....	461
Examples	463
41 SIP Profile Configuration	464
Introduction	465
SIP Profile Configuration Task List.....	465
Entering the Configuration Mode for a SIP Profile	465
Mapping from a SIP Disconnect Cause	465
Mapping to a SIP Cause	466
Mapping from a SIP Redirection Reason	466
Mapping to a SIP Redirection Code	466
SIP-Tunneling	466
Autonomous Transitioning for SIP	469
42 VoIP Profile Configuration	470
Introduction	471
VoIP Profile Configuration Task List	472
Creating a VoIP Profile	472
Configure Codecs	473
Configuring the Transparent-clearmode codec	475
Configuring the Cisco Versions of the G.726 Codecs	475
Configuring the AAL2-G.726-32k Codec	476
SDP prime Attribute	476
Configuring DTMF Relay	476
Configuring RTP Payload Types	477
Configuring RTP Payload Type for Transparent	478
Configuring RTP Payload Type for Transparent-cisco	478

Configuring RTP Payload Type for Transparent-clearmode	478
Configuring RTP Payload Types for the g726-32k and g726-32k-cisco Coders	478
Configuring RTP Payload Type for Cisco NSE	479
Configuring Cisco NSE for Fax	479
Configuring the Dejitter Buffer (advanced)	479
Enabling/Disabling Filters (advanced)	481
Configuring Fax Transmission	482
T.38 CED Retransmission	485
T.38 No-Signal Retransmission	485
Fax Bypass Method	485
Configuring Fax Failover	486
Configuring Modem Transmission	486
Modem Bypass Method	487
Configuring Packet Side Modem/Fax Answer Tone Detection	487
Disabling Fax/Modem Detection for Voice Calls	487
Media Processing	488
Configuring IP-IP Codec Negotiation	488
Configuring the Preferred Codec for Responses	488
RTP Statistics	489
Examples	489
Home Office in an Enterprise Network	489
Home Office with Fax	491
Soft Phone Client Gateway	493
43 PSTN Profile Configuration	495
Introduction	496
PSTN Profile Configuration Task List	496
Creating a PSTN Profile	496
Configuring the Echo Canceller	497
Configuring Output Gain	497
Configuring Input Gain	498
44 CS Context Overview	499
Introduction	500
CS Context Configuration Task List	501
Planning the CS Configuration	501
Configuring General CS Settings.....	503
Configuring the clock source	503
Debugging the clock source	504
Selecting PCM law compression	505
Configuring Call Routing.....	505
Creating and Configuring CS Interfaces	506
Specify Call Routing	506
Configuring Dial Tones	506
Configuring Voice Over IP Parameters.....	506

Configuring ISDN Ports	507
Configuring FXS Ports	507
Configuring a SIP VoIP Connection	507
Activating CS Context Configuration.....	508
Planning the CS Context	511
Configuring General CS Settings	512
Configuring Call Routing	512
Configuring VoIP Settings	514
Configuring BRI Ports	514
Configuring an SIP VoIP Connection	515
Activating the CS Context Configuration	515
Showing the Running Configuration	516
45 CS Interface Configuration	521
Introduction	522
CS Interface Configuration Task List	522
Creating and Configuring CS Interfaces.....	523
Configuring Call Routing.....	524
Configuring the Interface Mapping Tables	525
Configuring the Precall Service Tables.....	528
46 Tone Configuration	530
Introduction	531
Tone-set Profiles.....	531
Tone Configuration Task List	532
Configuring Call-Progress-Tone Profiles	532
Configure Tone-Set Profiles	533
Enable Tone-Set Profile	534
Show Call-Progress-Tone and Tone-Set Profiles	535
47 Authentication Service	537
Introduction.....	538
Authentication Service Configuration Task List	538
Creating an Authentication Service	538
Configuring a Realm	539
Configuring the Authentication Protocol	539
Creating Credentials	539
Configuration Examples.....	539
48 Location Service	540
Introduction.....	541
Location Service Configuration Task List.....	541
Creating a Location Service	541
Adding a Domain	541
Configuring Default Responsibility	542
Creating an Identity	542

Authentication outbound face	544
Authentication inbound face	545
Registration outbound face	546
Registration Priority	548
DNS Security Check	548
Support broken SIP proxy	549
NAT-traversal 'flows'	549
SIP B2BUA Dynamic Registration	550
Registration inbound face	550
Call outbound face	551
Configuring the Dynamic Registrar Use	552
Support broken SIP proxy	553
Call inbound face	554
Configuring SIP Transaction Timeout and Penalty Box	554
Creating an Identity Group	555
Inheriting from an Identity Group to an Identity	555
Configuring the Message Waiting Indication Feature for SIP	556
Subscription	556
Notification	557
Configuration	557
Message Waiting Indication through Call-Control	559
Show Location-Service	559
Configuration Examples	559
49 Call Router Configuration	561
Introduction	563
Call Router Configuration Task List	565
Map the Goals for the Call Router	565
Enable Advanced Call Routing on Circuit Interfaces	566
Configure General Call Router Behavior	566
Configure address completion timeout	566
Configure default digit collection timeout and terminating character	567
Configure Number Prefix for ISDN Number Types	568
Configure Call Routing Tables	568
Create a routing table	569
Called Party Number Routing Table	572
Regular expressions	572
Digit collection	573
Digit collection variants	574
Calling party number routing table	577
Number Type Routing Table	577
Numbering Plan Routing Table	578
Name Routing Table	579
IP Address Routing Table	579

URI Routing Table	580
Presentation Indicator Routing Table	580
Screening Indicator Routing Table	581
Information Transfer Capability Routing Table	582
Call-router Support for Redirecting Number and Redirect Reason	583
Time of Day Routing Table	583
Day of Week Routing Table	584
Date Routing Table	584
Deleting Routing Tables	584
Configure Routing Tables Dial Plan	585
Format	586
Configuration	586
Debugging	587
Status	587
Configure Mapping Tables	588
E.164 to E.164 Mapping Tables	593
Custom SIP URIs from Called-/Calling-e164 Properties	596
Other Mapping Tables	596
Deleting Mapping Tables	597
Creating Complex Functions	598
Deleting Complex Functions	599
Digit Collection & Sending-complete Behavior	600
Sending-complete	600
Ingress interface	600
Call-router	601
Egress interface	603
Creating Call Services	605
Creating a Hunt Group Service	605
Defining the Hunt-group Behavior for Inband Information	614
SIP handover on RTP loss with Hunt-Group	614
Creating a Distribution Group Service	616
Distribution-Group Min-Concurrent Setting	618
Call-router 'limiter' Service	618
Priority Service	619
CS Bridge Service—'VoIP Leased Line'	620
Configuring the Service Second-dialtone	623
Deleting Call Services	624
Activate the Call Router Configuration	624
Test the Call Router Configuration	625
Configuring Partial Rerouting	631
Call reroute	631
Enable rerouting requests on ISDN.	631
Enable emission of rerouting requests on ISDN.	632
Enable sending of "302 moved temporary" message on SIP.	632

Allow push-back	632
Enable push-back – aaa service	633
Enable push-back – bridge service	633
Enable push-back – distribution-group service	633
Enable push-back – hunt group service	633
Enable push-back – limiter service	633
Enable push-back – priority service	633
50 Global SIP Configuration	634
Introduction	635
Enabling SIP Client ‘rport’ Support	635
Configuring the SIP User-Agent Header	635
Configuring SIP Peer Flood Blocking.....	635
Limit Packets to Prevent SIP Overload Condition.....	636
Locking DNS Records for SIP Requests	636
SIP Request URI Length Limitation.....	637
51 SIP Overload Configuration	638
Introduction	639
System Overview	639
Default Behavior	640
Mode of operation	640
SIP overload configuration task list.....	642
Disable SIP message rejection	642
Change the list of SIP messages rejected in an overload situation.....	643
Reset the SIP overload behavior to the system defaults	643
Display the behavior of SIP in an overload situation.....	644
Debug the behavior of SIP in an overload situation	646
52 SIP Interface Configuration	647
Introduction	648
SIP Interface Configuration Task List	649
Binding the Interface to a SIP Gateway	649
Configuring a Remote Host	650
Managing trusted hosts	650
Configuring a Local Host (Optional)	650
Using an Alternate VoIP Profile (Optional)	651
Using an Alternate SIP Profile (Optional)	651
Using an Alternate Tone-Set Profile (Optional)	652
Configuring Early Call Connect/Disconnect (Optional)	652
Enable/Disable Early-proceeding on SIP Interface	653
Early Media Behavior	653
Configuring Address Translation (Optional)	653
Mapping call-control properties in SIP headers	653
Mapping SIP headers to call-control properties	654
Configuring ISDN redirecting number tunneling over SIP	654

Enabling SIP RFC privacy, asserted-identity, & preferred-identity headers (RFC 3323/3325)	655
Updating caller address parameters	655
SIP diversion header	656
Transmit Direction	656
Receive Direction	657
SIP Calling Party Number Handling	657
SIP REFER Transmission (& ISDN Explicit Call Transfer support)	657
AOC Over SIP (Optional)	658
Enabling the Session Timer (Optional)	659
Enabling the SIP Penalty-box Feature (Optional)	660
Initiating a New SIP Session for Redirected SIP Calls (Optional)	661
Rerouting Calls from SIP (Optional)	661
Configuring the SIP Hold Method (Optional)	661
Configuring SIP Overlap Dialing (Optional)	661
Configuring PRACK for Reliable Provisional Responses (optional)	662
Configuring History-Info in SIP	663
Enabling NAT Traversal for SIP INVITE Messages	664
Accepting Re-invite After Reboot	664
53 SIP Survivability	665
Introduction	666
Background	666
Operation Principle	666
SIP-Survivability Configuration Tasks	668
Set up an Ethernet bridge	669
Create a Location Service	669
Create a SIP Gateway	670
Configure Call-Routing	670
Configure the SIP-Survivability Service	671
Configure the SIP-Survivability Agent	671
Essential Configuration	672
Optional commands for tuning the observation of SIP messages	672
Optional commands for tuning the observation of DNS messages	674
Inspect the SIP-Survivability Status	677
Debugging the SIP Survivability Agent	680
54 TLS Configuration for SIP	682
Introduction	684
About Transport Layer Security (TLS)	684
TLS configuration task list	684
Install license	684
Configure URI scheme	684
URI Scheme for Calls	685
SIP	685
SIPS	685

Transparent	685
URI Scheme for Registration	686
SIP	686
SIPS	686
URI Scheme for Re-routed Calls	686
Rerouted	686
Original	686
URI Scheme for Transferred Calls	687
Configure Time	687
Configure Public-Key Infrastructure (PKI)	687
About TLS Profiles	687
Basic Operation Principle of TLS	688
TLS without authentication (default behavior)	688
TLS authentication with self-signed certificates	688
TLS Authentication with 3rd Party Signed Certificates	689
TLS Profile Default	689
Configure TLS Profile	689
Creating a TLS profile and enter configuration mode	689
Private key	690
Own-certificate chain	690
Trusted Certificates	691
Authentication	692
Cipher Suites	693
Compression	694
Protocol	694
Configure TLS profile usage	694
Configure TLS transport	694
Configure transport enforcement or fallback	695
Configure Non-Default TLS Port Usage	697
Configure SRTP	698
Security consideration: Why you should use mutual TLS authentication	699
IP does not Re-use Connections for Requests in Different Directions	699
Two Scenarios of TLS Authentication Vulnerability	700
Conditions for the Patton device to Re-use Connections	700
Caveat for TLS Profile	701
Showing TLS profile	701
Troubleshooting	702
Check License	702
Check time	702
Check Certificates	703
Check TLS Profile Status	704
Check SIP Gateway Status	706
Check Server Name	706

55	TLS Configuration for Lync	708
	Introduction.....	709
	Configuration options: mutual vs. server authentication.....	709
	Task List.....	709
	1. Prepare the Lync server.....	710
	Option A: Prepare the Lync server for mutual authentication	710
	Lync CA: Create an authentication template for mutual authentication	711
	Lync CA: Activate the new template on the Certification Authority (CA)	712
	Lync: Issue a certificate for the Lync server with the mutual authentication template	712
	Lync: Configure the Lync server	712
	Option B: Prepare the Lync server for server authentication only	713
	Lync: Issue a certificate for the Lync server with the server authentication template	713
	Lync: Configure the Lync server	713
	2. Configure basic settings on the Patton device running Trinity.....	713
	Trinity: Configure DNS	714
	Trinity: Configure time	714
	3. Issue a certificate on the Lync server and import it on the Patton device running Trinity.....	714
	Option A: Export certificate request from Trinity	714
	Trinity: Generate private key	715
	Trinity: Generate certificate request	715
	Trinity: Export certificate request	716
	Lync CA: Sign certificate request	716
	Trinity: Import own certificate	718
	Trinity: Import trusted certificate	718
	Option B: External certificate administration	719
	Lync CA: External certificate creation	719
	Trinity: Import private key	719
	Trinity: Import own certificate	720
	Trinity: Import trusted certificate	720
	4. Configure TLS on the Patton device running Trinity.....	720
	Configure Trinity TLS profile	720
	Option A: Configure TLS profile with mutual authentication	720
	Option B: Configure TLS profile with server authentication	721
	Trinity: Configure SIP with TLS	721
	Final Configuration on Trinity device.....	722
56	Secure SIP Applications	725
	Introduction.....	726
	TLS/SRTP encryption without TLS authentication	726
	Configuration without TLS/SRTP	726
	Tasks to Configure TLS/SRTP.....	728
	Configure URI-scheme	728
	Enable TLS on the SIP gateway	728
	Enable SRTP on the VoIP profile	729

Configuration with TLS/SRTP	729
TLS/SRTP encryption with mutual TLS authentication (MTLS) based on exchanged, locally-generated, self-signed certificates	730
Tasks to create and exchange self-signed certificates.....	730
Generate private key	731
Generate certificate request	731
Self-sign the certificate request	731
Exchange certificates	731
Configure the TLS profile	731
Configuration with TLS/SRTP and mutual TLS authentication	732
TLS/SRTP encryption with mutual TLS authentication (MTLS) in a Microsoft Lync environment.....	733
57 Secure Real-Time Protocol (SRTP) Configuration	734
Introduction.....	735
About SRTP.....	735
SRTP configuration task list.....	735
License	735
Information about using DSP channels	735
Configure secure call signaling	736
Configure RTP security	736
Protection against key compromising through forking	738
Protection against key compromising through call forward	738
58 SIP Security	740
Introduction.....	742
Transport Layer Security (TLS).....	742
TLS configuration task list	742
Install license	742
Configure URI scheme	743
URI Scheme for Calls	743
SIP	743
SIPS	743
Transparent	743
URI Scheme for Registration	744
SIP	744
SIPS	744
URI Scheme for Re-routed Calls	744
Rerouted	744
Original	744
URI Scheme for Transferred Calls	745
Configure Time	745
Configure Public-Key Infrastructure (PKI)	745
About TLS Profiles	745
Basic Operation Principle of TLS	746
TLS without authentication (default behavior)	746

TLS authentication with self-signed certificates	746
TLS Authentication with 3rd Party Signed Certificates	747
TLS Profile Default	747
Configure TLS Profile	747
Creating a TLS profile and enter configuration mode	747
Private key	748
Own-certificate chain	748
Trusted Certificates	749
Authentication	750
Cipher Suites	751
Compression	752
Protocol	752
Configure TLS profile usage	752
Configure TLS transport	752
Configure transport enforcement or fallback	753
Configure Non-Default TLS Port Usage	755
Configure SRTP	756
Security consideration: Why you should use mutual TLS authentication	757
IP does not Re-use Connections for Requests in Different Directions	757
Two Scenarios of TLS Authentication Vulnerability	759
Conditions for the Patton device to Re-use Connections	759
Caveat for TLS Profile	760
Showing TLS profile	760
Troubleshooting	760
Check License	761
Check time	761
Check Certificates	761
Check TLS Profile Status	762
Check SIP Gateway Status	764
Check Server Name	764
Secure Real-Time Protocol (SRTP) Configuration	766
SRTP configuration task list	766
License	766
Information about using DSP channels	766
Configure secure call signaling	767
Configure RTP security	767
Protection against key compromising through forking	769
Protection against key compromising through call forward	769
59 SIP Call-router Services	771
Introduction	772
SIP Conference-service	772
SIP Conference-service Configuration Task List	772
Entering conference-service configuration mode	772

Configuring the call routing destination	772
Configuring the conference server	773
SIP Location-service	773
SIP Location-service Configuration Task List	774
Entering SIP location-service configuration mode	774
Binding a location service	775
Configuring multi-contact behavior	775
Configuring the hunt timeout	775
60 Context SIP Gateway Overview	777
Introduction	778
Context SIP Gateway Configuration Task List	779
Creating a Context SIP Gateway	779
Creating a Transport Interface	779
Configuring the IP Binding	780
Configuring a Spoofed Contact Address	780
Contact-Header	781
Via-Header	781
Nat-Address	781
Binding Location Services	782
SIP Trusted Host Behavior	782
Setting a Traffic Class	782
Configuring Quality of Protection in SIP Authentication	782
Enabling/Disabling the Context SIP Gateway	783
SIP notify check-sync event	783
Troubleshooting	784
Show Status Information	784
Debug Commands	784
Configuration Examples	784
Example 1	784
Example 2	785
Example 3	785
Applications	785
Outbound Authentication	785
Inbound Authentication	787
Outbound Registration	788
Inbound Registration	789
B2B User Agent with Registered Clients	790
61 ISDN Overview	793
Introduction	794
ISDN Reference Points	794
Possible Patton device Port Configurations	795
ISDN UNI Signaling	795
ISDN Configuration Concept	797

ISDN Layering	797
62 ISDN Configuration	798
Introduction	799
ISDN Configuration Task List	799
Enter Q.921 Configuration Mode	799
Configuring Q.921 Parameters	800
Configuring tei Assignment Procedure on ISDN	800
Configuring Q.931 Encapsulation	801
Enter Q.931 Configuration Mode	801
Configuring Q.931 Parameters	801
Configuring a Channel Identifier on ISDN PRI	803
Configuring Q.931 Application Protocol Encapsulation	803
Debugging ISDN	803
ISDN Configuration Examples	804
63 ISDN Interface Configuration	806
Introduction	807
ISDN Interface Configuration Task List	807
Configuring DTMF Dialing (optional)	808
Configuring an Alternate PSTN Profile (optional)	808
Configuring Ringback Tone on ISDN User-side Interfaces	809
Configuring Call Waiting (optional)	809
Disabling Call Waiting on ISDN DSS1 Network Interfaces	809
Configuring Call-Hold on ISDN Interfaces	810
Enabling Display Information Elements on ISDN Ports	810
Configurable calling party or facility IE on ISDN	810
Configuring Date/Time Publishing to Terminals (optional)	810
Sending the Connected Party Number (COLP) (optional)	811
Enabling Sending of Progress-indicator on ISDN (optional)	811
Defining the 'network-type' in ISDN Interfaces	811
ISDN Explicit Call Transfer Support (& SIP REFER Transmission)	812
ISDN Advice of Charge Support	814
ISDN DivertingLegInformation2 Facility	817
Transmit direction	817
Receive direction	818
T1 Caller-Name Support	818
Caller-Name Facility Format on QSIG	820
Format Examples	820
Tunneling of ISDN UUI1 Information Over SIP	820
Configuring the Message Waiting Indication Feature for ISDN	821
Configuring the Release Tone on ISDN DSS1 Network Interfaces	821
Configuring the ISDN User-side Timer T304	822
64 FXS Interface Configuration	823
Introduction	825

Configuring Conference Routing	829
Configuring the Ringing-Cadence	830
Creating a new Ringing-Cadence Profile	831
Populating the Ringing-Cadence	831
Flushing the Ringing-Cadence	832
Displaying the Configuration of Ringing-Cadence Profiles	832
Using the Ringing-Cadence Profile on FXS interfaces	833
Configuring Distinctive-Ringing	833
Configuring the Message Waiting Indication Feature for FXS	835
Configuration	836
Configuring Supplementary Services	836
Configured Service Activation	837
Dynamic Service Activation/Deactivation/Interrogation	840
Service Invocation	841
Creating a new FXS Supplementary-Services Profile	841
Using the FXS Supplementary-Services Profile on FXS interfaces	841
Configuring Flash-Hook Processing	842
Call Hold	844
Subscription/Activation	844
Invocation (Mute)	845
Invocation (Additional Call)	846
Call Waiting	848
Subscription	848
Activation	849
Invocation	849
Call Waiting Reminder Ring	850
Subscription/Activation	851
Invocation	851
Drop Passive Call	852
Subscription/Activation	853
Invocation	853
Drop Active Call	855
Subscription/Activation	855
Invocation	855
Call Toggle	857
Subscription/Activation	857
Invocation	857
Call Transfer	859
Subscription/Activation	860
Invocation	860
Conference	862
Subscription/Activation	862
Invocation	864
Call Park	866

Subscription/Activation	866
Invocation	866
Advice-of-Charge (AOC) Tax-Pulses	868
Converting AOC Information into Tax-Pulses	868
Local Tax-Pulse Generation	870
65 PRI Port Configuration	872
Introduction	873
Terminology	873
PRI Port Configuration task List	873
Enable/Disable PRI Port	873
Configuring PRI Port-type	874
Configuring PRI Clock-mode	874
Configuring PRI Line-code	874
Configuring PRI Framing	875
Configuring PRI Line-Build-Out (E1T1 in T1 mode only)	875
Configuring PRI Application Mode (E1T1 only)	875
Configuring PRI LOS Threshold (E1T1 only)	876
Configuring PRI Encapsulation	876
PRI Debugging	876
66 E1/T1 Port Configuration	878
Introduction	879
Patton support headquarters in the USA	879
Alternate Patton support for Europe, Middle East, and Africa (EMEA)	879
67 BRI Port Configuration	880
Introduction	881
BRI Port Configuration task List	881
Enable/Disable BRI Port	881
Configuring BRI Clock-mode	881
Configuring BRI Power-Feed	882
Configure BRI Line-Termination	882
Configuring BRI Encapsulation	882
BRI Debugging	882
BRI Configuration Examples	883
Example 1: ISDN with auto clock/uni-side settings	883
Example 2: ISDN with manual clock/uni-side settings	883
68 FXS Port Configuration	885
Introduction	886
Shutdown and Enable FXS Ports	888
Bind FXS Ports to FXS Interfaces in the CS Context	889
Configure a Subscriber Number	889
Configure Phone-Specific Parameters	890
Plan your FXS profiles	891

Use an existing FXS profile on an FXS port	894
Create, modify, reset, or delete an FXS profile	894
Configure the Electrical Standard	895
Configure the Caller-ID Presentation	896
Configure Connect and Disconnect Signals	897
Configure Tax-Pulse Modulation	899
Configure Message-Waiting Indication (MWI) Feature for FXS	900
Displaying the Configuration and State of an FXS port.....	901
69 Debug and Monitoring	906
Introduction	907
Debugging Strategy	907
Verifying IP Connectivity.....	908
Network Sniff Tool	908
Debugging Call Signaling.....	908
Debugging ISDN Signaling	909
Debugging SIP Signaling	909
Verify an incoming call	910
Verify an outgoing call	910
Debugging FXS Signaling	910
Verify an incoming call	911
Verify an outgoing call	913
Using Trinity's Internal Call Generator	914
Debugging the SIP Stack.....	914
Debugging Voice Data	915
How to Submit Trouble Reports to Patton	917
What is PacketSmart?	917
PacketSmart in Trinity	918
Configuring a PacketSmart Profile	918
What can I do with PacketSmart?	919
70 Contacting Patton for Assistance	920
Introduction.....	921
Contact information.....	921
Contacting Patton Technical Services for Free Support	921
Warranty Service and Returned Merchandise Authorizations (RMAs).....	921
Warranty coverage	921
Out-of-warranty service	922
Returns for credit	922
Return for credit policy	922
RMA numbers	922
Shipping instructions	922
A Trinity Architecture Terms and Definitions	923
Introduction.....	924

B	Command Summary	929
	Introduction.....	930
	New Configuration Commands	931
	Other	931
	Show help	931
	Show command history	931
	Restart system	931
C	Glossary of Terms	932
	List of terms	933

List of Figures

1	Basic system (abstract) model	53
2	Typical carrier network application with a Patton device	54
3	Typical enterprise network with a Patton device	55
4	Typical LAN telephony system with a Patton device gateway	56
5	Configuration concept overview	58
6	Setup for initial configuration via the console port	69
7	Sample configuration file	90
8	Local memory regions	91
9	Remote memory regions for Trinity	93
10	How licenses may be made available to Patton devices.	99
11	Install licenses permanently on the device	100
12	Device leases floating licenses from a license server	100
13	Connected Devices are grouped to Organizations and Sub-Organizations	107
14	Authentication procedure with a TACACS+ server	128
15	How to use AAA methods and AAA profiles	128
16	The Event-Programming System is organized in three layers	151
17	Expression Tree	153
18	Time Diagram of the DEBOUNCE Functions	176
19	Time Diagram of the Functions ONCE, STABLE, and DELAY	176
20	State Profile	178
21	State Transition Hysteresis	178
22	Binding of an Ethernet port to an IP interface	199
23	NETGEAR 341U Manager screen	212
24	VLAN example 1	222
25	VLAN example 2	223
26	Using traffic filters to prevent traffic from being routed to a network	225
27	ADSL/VDSL (IPoE)	254
28	ADSL/VDSL (PPPoE)	255
29	ADSL/VDSL (IPoEoA)	257
30	ADSL/VDSL (PPPoEoA)	257
31	Spanning tree configuration	268
32	PPP configuration overview	271
33	IP context and related elements	281
34	Processing order of IP services attached to an IP interface	283
35	Static route example	303
36	Policy-routing observation points	305
37	Whole ip interface assignment	307
38	Explicit traffic assignment	309
39	Classifier Directions	311
40	Site-to-site IPsec tunnel being established between a branch office and the main office	325
41	Point-to-point	330
42	Point-to-Multipoint	330
43	GRE Tunnel across a WAN	341
44	Diagram of sample configuration.	344
45	Load balancing and failover example	358
46	After all my WAN links go down, they never come back up	368
47	Dynamic NAPT	374

48	Static NAT	374
49	Dynamic NAT	375
50	Static NAT	375
51	DHCP-client and DHCP-server	381
52	DNS relay diagram	389
53	ManageEngine MibBrowser displaying some of the System Group objects	398
54	ManageEngine MibBrowser Settings Button on the Toolbar	403
55	ManageEngine TrapViewer displaying received traps	404
56	ManageEngine Trap Details window of TrapViewer	405
57	PKI Architecture	410
58	Symmetric Encryption	410
59	Private/Public Key Generation	411
60	Asymmetric Encryption	411
61	Certificate Enrollment Process	412
62	Self-signed Certificate Process	413
63	Conceptual view on the classifier; it groups packet flows of the same service	424
64	Mapping TOS/CoS to traffic-class and vice-versa	425
65	Using traffic filters to prevent traffic from being routed to a network	442
66	Deny a specific subnet on an interface	448
67	Deny traffic between two interfaces	449
68	Locations in the routing path where packets may be classified	452
69	VoIP profile association	471
70	DTMF Relay	477
71	Jitter and dejitter buffer	480
72	Adaptive versus static dejitter buffer	481
73	Multiple tandem and sequential post filtering	481
74	Fax relay and Fax bypass	483
75	Home office in an enterprise network	490
76	PSTN profile association	496
77	Echo Cancellation	497
78	Applying output gain	497
79	Applying input gain	498
80	CS context configuration components	500
81	Remote office in an Enterprise network	502
82	Direct call routing from one Patton device to another	505
83	Patton device in an Enterprise network	510
84	CS Configuration	511
85	CS interfaces on the CS context	522
86	Incoming call passing an interface mapping table	527
87	Call passing an input and an output mapping table	528
88	Assign tone-sets to a PSTN interfaces	532
89	Direct call routing vs. advanced call routing	564
90	Routing table outline	569
91	Mapping table outline	588
92	Mapping table examples	592
93	Hunt group service	606
94	Distribution group service	616
95	Distribution group service examples	617
96	'Limiter' service diagram	618
97	Priority service diagram	619
98	CS Bridge service—'VoIP Leased Line' diagram	621

99	Bridge services diagram	622
100	Call routing example network	627
101	CS context and call router elements	629
102	The SIP overload behavior uses the OVERLOAD state profile	639
103	SIP Overload configuration and message flow diagram	641
104	State-Based SIP Access Control Lists (ACL)	642
105	SIP interfaces on the CS context	648
106	The Patton device contains a survivability agent software component.	667
107	TLS Connection Establishments and Re-use in SIP	699
108	Patton device secures a SIP connection between two SIP soft-phones.	727
109	Patton devices secures a SIP connection between two SIP soft-phones.	730
110	TLS Connection Establishments and Re-use in SIP	758
111	Registration and Lookup	774
112	Routing Architecture	778
113	ISDN reference points	794
114	ISDN signaling side	795
115	Integration of ISDN access lines	796
116	ISDN layering model	797
117	PBX connected to ISDN port 1/0	805
118	ISDN interfaces on the CS context	807
119	Example SIP network connecting two device to give a home office access to the CO PBX	813
120	FXS-Interface configuration components	825
121	Local vs. Relayed Flash-Hook Processing	844
122	Call-Hold Supplementary-Service Invocation (no consultation call allowed)	846
123	Call-Hold Supplementary-Service Invocation (additional consultation call allowed)	847
124	Call-Waiting Supplementary-Service Invocation	850
125	Call-Waiting-Reminder-Ring Supplementary-Service Invocation from Call Hold	851
126	Call-Waiting-Reminder-Ring Supplementary-Service Invocation from Call Waiting	852
127	Drop-Passive-Call Supplementary-Service Invocation from Call Hold	854
128	Drop-Passive-Call Supplementary-Service Invocation from Call Waiting	854
129	Drop-Active-Call Supplementary-Service Invocation from Call Hold	856
130	Drop-Active-Call Supplementary-Service Invocation from Call Waiting	856
131	Call-Toggle Supplementary-Service Invocation from Call Hold	858
132	Call-Toggle Supplementary-Service Invocation from Call Waiting	859
133	Call-Transfer Supplementary-Service Invocation from Call Hold	861
134	Call-Transfer attempt from Call Waiting; triggers Call Waiting Reminder Ring	862
135	Conference Supplementary-Service Invocation from Call Hold	865
136	Call-Park Supplementary-Service Invocation	867
137	Advice-of-Charge to Tax-Pulse Conversion and Tax-Pulse Generation	869
138	FXS-Port configuration components	886
139	What PacketSmart commands look like in a Trinity config	919
140	Output of the command "show profile packetsmart":	919
141	EBNF syntax	930

List of Tables

1	General conventions	48
2	Essential System Variables	151
3	Regular Expression Tokens	160
4	Error Values	163
5	Logical Operators	164
6	Bitwise Operators	165
7	Arithmetic Operators	166
8	Comparison Operators	166
9	Operator Precedence	168
10	Logical Functions	169
11	Bitwise Functions	170
12	Arithmetic Functions	170
13	Comparison Functions	172
14	Set Functions	173
15	Time/Date Functions	173
16	Temporal Functions	174
17	Cellular modems supported by Trinity	206
18	Payload Rate Configuration Overview	251
19	Creating Bridge Group	261
20	Setting Various Bridge Options	261
21	Enable Filters on the Bridge-Group	262
22	Set STP Options	262
23	Configure VLANs	262
24	Bind Resources to the Bridge-Group	262
25	Show Bridge-Group Configuration	263
26	Details available in the Trap Details window	405
27	Command Line Syntax	461
28	ISDN number types	568
29	Routing table types	569
30	Wildcard symbols used as keys in E.164 tables (calling-e164, called-e164)	572
31	Examples of using wildcard symbols	573
32	Mapping table types	589
33	Hunt group drop causes	608
34	Default SIP overload behavior	640
35	Default keypad sequences to invoke FXS supplementary services	841
36	Default settings for built-in and manually-created FXS profiles	891
37	Country-Codes to select the FXS electrical standard	895
38	Caller-ID standards	897
39	Connect and disconnect signals	898
40	Tax-Pulse Modulation Methods	899
41	FXS Port Fields	902

About this guide

The objective of this *Trinity Release 3.12 Command Line Reference Guide* is to provide information concerning the syntax and usage of the command set.

This section describes the following:

- Who should use this guide (see “[Audience](#)”)
- How this document is organized (see “[Structure](#)”)
- Typographical conventions and terms used in this guide (see “[Typographical conventions used in this document](#)” on page 48)

Audience

This guide is intended for the following users:

- System administrators who are responsible for installing and configuring networking equipment and who are familiar with the Trinity.
- System administrators with a basic networking background and experience, but who might not be familiar with the Trinity.
- Operators
- Installers
- Maintenance technicians

How to read this guide

Trinity is a complex and multifaceted operating system. Without the necessary theoretical background you will not be able to understand and use all the features available. Therefore, we recommend reading at least the chapters listed below to get a general idea about Trinity and the philosophy of contexts used for IP and circuit switching related configuration.

- Appendix A, “[Trinity Architecture Terms and Definitions](#)” on page 923 contains the terms and their definitions that are used throughout this *Trinity Release 3.x Command Line Reference Guide*.
- Chapter 1, “[System Overview](#)” on page 51 provides an overview of the main elements of a Trinity system.

Structure

This guide contains the following chapters and appendices:

- Chapter 1, “[System Overview](#)” on page 51 provides an overview of the main elements of a Trinity system.
- Chapter 2, “[Configuration Concepts](#)” on page 57 introduces basic Trinity configuration concepts.
- Chapter 3, “[Command Line Interface \(CLI\)](#)” on page 62 gives an overview of the CLI and the basic features that enable you to navigate the CLI and edit commands effectively.
- Chapter 4, “[Accessing the CLI](#)” on page 67 describes the procedures for entering Trinity commands via the command line interface (CLI) to obtain help, to change operator mode, and to terminate a session.
- Chapter 5, “[Creating CLI Action Scripts](#)” on page 79 describes how to create CLI action scripts that execute on specific events (for example, link down on the IP interface).

- Chapter 6, “[System Image Handling](#)” on page 84 describes how to load and maintain system images and driver software.
- Chapter 7, “[Configuration File Handling](#)” on page 88 describes how to upload and download configuration files from and to a Patton device.
- Chapter 8, “[System Licensing](#)” on page 98 describes how to configure system licensing in Trinity.
- Chapter 9, “[AAA Configuration](#)” on page 126 provides an overview of configuring TACACS+.
- Chapter 10, “[Basic System Management](#)” on page 135 describes parameters that report basic system information to the operator or administrator, and their configuration.
- Chapter 11, “[Programmable System-Event Configuration](#)” on page 149 describes how to use programmable system events in Trinity.
- Chapter 12, “[Alarm Management](#)” on page 182 describes how to configure the alarm subsystem.
- Chapter 13, “[Auto Provisioning of Firmware and Configuration](#)” on page 185 provides an overview of Trinity’s Auto Provisioning capabilities and tasks involved to configure it.
- Chapter 14, “[Ethernet Port Configuration](#)” on page 197 provides an overview of Ethernet ports and describes the tasks involved in their configuration through Trinity.
- Chapter 15, “[Configuring a Cellular Modem](#)” on page 205 describes how to configure a cellular modem.
- Chapter 16, “[Hardware Switching](#)” on page 213 provides an overview of devices with an internal switch for connecting with external ports.
- Chapter 17, “[DSL Port Configuration](#)” on page 248 provides an overview of the DSL ports, their characteristics and the tasks involved in the configuration.
- Chapter 18, “[Context Bridge](#)” on page 260 describes how to configure bridging with Ethernet ports.
- Chapter 19, “[Spanning Tree Configuration](#)” on page 264 provides an overview of how to configure classic, rapid, and multiple spanning tree protocol.
- Chapter 20, “[PPP Configuration](#)” on page 270 describes how to configure the point-to-point protocol over different link layers.
- Chapter 21, “[IP Context Overview](#)” on page 280 outlines Trinity *Internet protocol* (IP) context, together with its related components.
- Chapter 22, “[IP Interface Configuration](#)” on page 289 provides a general overview of Patton device interfaces and describes the tasks involved in their configuration.
- Chapter 23, “[IP Routing](#)” on page 300 provides an overview of IP routing and describes the tasks involved in configuring static IP routing in Trinity.
- Chapter 24, “[Internet Protocol Security \(IPsec\) Configuration](#)” on page 314 gives an overview of the Internet Protocol Security (IPsec), and describes how to use IPsec commands to configure Trinity.
- Chapter 25, “[OpenVPN Configuration](#)” on page 329 describes how to configure an OpenVPN tunnel in Trinity.
- Chapter 26, “[Generic Routing Encapsulation \(GRE\) Configuration](#)” on page 340 describes how to configure generic routing encapsulation (a GRE tunnel) in Trinity.
- Chapter 27, “[Border Gateway Protocol \(BGP\) Configuration](#)” on page 345 gives an overview of the Border Gateway Protocol (BGP), and describes how to use BGP commands to configure Trinity.
- Chapter 28, “[Load Balancing and Failover Configuration](#)” on page 357 describes how to implement load balancing and failover.

- Chapter 29, “[Fast-Path](#)” on page 369 explains how to use the new Fast Path feature to speed-up the routing performance of Trinity devices.
- Chapter 30, “[NAT/NAPT Configuration](#)” on page 372 provides a general overview of the network address port translation and describes the tasks involved in its configuration.
- Chapter 31, “[DHCP Configuration](#)” on page 380 provides an overview of the *Dynamic Host Configuration Control Protocol* (DHCP) and describes the tasks involved in its configuration.
- Chapter 32, “[DNS Configuration](#)” on page 386 describes how to configure the *domain name system* (DNS) component.
- Chapter 33, “[SNTP Client Configuration](#)” on page 390 describes how to configure a *simple network time protocol* (SNTP) client.
- Chapter 34, “[SNMP Configuration](#)” on page 393 provides overview information about the *simple network management protocol* (SNMP) and describes the tasks used to configure those of its features supported by Trinity.
- Chapter 35, “[Public-Key Infrastructure \(PKI\)](#)” on page 408 provides an overview on how to set up the public-key infrastructure (PKI) on a Patton device.
- Chapter 36, “[Quality of Service \(QoS\) Overview](#)” on page 423 provides an overview of the core principles of Trinity’s Quality-of-Service architecture.
- Chapter 37, “[Profile Service-Policy Configuration](#)” on page 427 describes how to use and configure service-policy profiles.
- Chapter 38, “[Access Control List Configuration](#)” on page 440 provides an overview of IP *Access Control Lists* (ACLs) and describes the tasks involved in configuring them.
- Chapter 39, “[Classifier Configuration](#)” on page 450 provides an overview of Trinity’s packet classifier and describes the tasks involved in its configuration.
- Chapter 40, “[Service Policy Configuration](#)” on page 458 describes how to use and configure service-policy profiles.
- Chapter 40, “[Packet Matching](#)” on page 458 lists the criteria that may be used to match packets and specifies the command line syntax.
- Chapter 41, “[SIP Profile Configuration](#)” on page 464 describes the SIP profile, which specifies disconnect cause mappings from SIP codes to Q.931 causes, and vice versa.
- Chapter 42, “[VoIP Profile Configuration](#)” on page 470 provides an overview of VoIP profiles, and describes how they are used and the tasks involved in VoIP profile configuration.
- Chapter 43, “[PSTN Profile Configuration](#)” on page 495 provides an overview of PSTN profiles, and describes how they are used and the tasks involved in PSTN profile configuration.
- Chapter 44, “[CS Context Overview](#)” on page 499 provides an overview of the circuit-switching (CS) context and associated components, and describes the tasks involved in its configuration.
- Chapter 45, “[CS Interface Configuration](#)” on page 521 provides an overview of interfaces in the CS context and describes the tasks involved in their configuration.
- Chapter 46, “[Tone Configuration](#)” on page 530 provides an overview of call-progress-tone profiles and tone-set profiles, and describes the tasks involved in their configuration.
- Chapter 47, “[Authentication Service](#)” on page 537 describes how to configure authentication services in Trinity.
- Chapter 48, “[Location Service](#)” on page 540 describes how to configure location services in Trinity.

- Chapter 49, “[Call Router Configuration](#)” on page 561 provides an overview of call router tables, mapping tables and call services and describes the tasks involved in configuring the call router in Trinity.
- Chapter 50, “[Global SIP Configuration](#)” on page 634 describes the Trinity commands available under the global SIP configuration mode.
- Chapter 51, “[SIP Overload Configuration](#)” on page 638 describes how to configure the Trinity device's behavior in case of an overload situation.
- Chapter 52, “[SIP Interface Configuration](#)” on page 647 provides an overview of SIP interfaces used by context SIP gateways and describes the specific tasks involved in their configuration.
- Chapter 53, “[SIP Survivability](#)” on page 665 describes how to set up the SIP survivability agent that is built into the Patton device.
- Chapter 54, “[TLS Configuration for SIP](#)” on page 682 provides an overview of Trinity’s Transport Layer Security (TLS) capabilities for SIP and describes the tasks involved in configuring it.
- Chapter 55, “[TLS Configuration for Lync](#)” on page 708 explains how to integrate a Patton VoIP gateway to a Microsoft Lync network.
- Chapter 56, “[Secure SIP Applications](#)” on page 725 discusses different Transport Layer Security (TLS) and Secure Real-Time Protocol (SRTP) scenarios and provides the configuration snippets to set up the Patton device for them.
- Chapter 57, “[Secure Real-Time Protocol \(SRTP\) Configuration](#)” on page 734 provides an overview of Trinity’s SRTP capabilities and describes the tasks involved in configuring it.
- Chapter 58, “[SIP Security](#)” on page 740 provides an overview of Trinity’s *Transport Layer Security* (TLS) capabilities for SIP and describes the tasks involved in configuring it.
- Chapter 59, “[SIP Call-router Services](#)” on page 771 contains the description of all SIP specific call router services, which are only available if the software includes the SIP component.
- Chapter 60, “[Context SIP Gateway Overview](#)” on page 777 provides an overview of the context SIP gateway.
- Chapter 61, “[ISDN Overview](#)” on page 793 provides an overview of ISDN ports and describes the tasks involved in configuring ISDN ports.
- Chapter 62, “[ISDN Configuration](#)” on page 798 describes the configuration of the Q.921 and Q.931 protocol and how to bind the ISDN protocol to an application like the Call Control.
- Chapter 63, “[ISDN Interface Configuration](#)” on page 806 provides an overview of ISDN interfaces, and the tasks involved in their configuration.
- Chapter 64, “[FXS Interface Configuration](#)” on page 823 provides an overview of FXS interfaces, and the tasks involved in their configuration.
- Chapter 65, “[PRI Port Configuration](#)” on page 872 provides an overview of the PRI (Primary Rate Interface) ports, their characteristics and the tasks involved in the configuration.
- Chapter 66, “[E1/T1 Port Configuration](#)” on page 878 provides an overview of the E1/T1 ports, their characteristics and the tasks involved in the configuration.
- Chapter 67, “[BRI Port Configuration](#)” on page 880 provides an overview of the BRI (Basic Rate Interface) ports, their characteristics and the tasks involved in their configuration.
- Chapter 68, “[FXS Port Configuration](#)” on page 885 provides an overview of POTS signaling (Plain Old Telephone System) and describes the tasks involved in configuring FXS ports in Trinity.

- Chapter 69, “[Debug and Monitoring](#)” on page 906 describes how to debug VoIP sessions, including the signaling part and the voice data path part (speech, fax, and modem connectivity).
- Chapter 70, “[Contacting Patton for Assistance](#)” on page 920 describes how to contact Patton technical support for assistance, and contains information about the warranty and obtaining a return merchandise authorization (RMA).
- Appendix A, “[Trinity Architecture Terms and Definitions](#)” on page 923 contains terms and definitions that are used in this *Trinity Software Configuration Guide*.
- Appendix B, “[Command Summary](#)” on page 929 is a command reference.
- Appendix C, “[Glossary of Terms](#)” on page 932 is the glossary of terms.

Precautions

The following are used in this guide to help you become aware of potential problems:

Note A note presents additional information or interesting sidelights.



The alert symbol and IMPORTANT heading calls attention to important information.

Typographical conventions used in this document

This section describes the typographical conventions and terms used in this guide.

General conventions

In this guide we use certain typographical conventions to distinguish elements of commands and examples. In general, the conventions we use conform to those found in IEEE POSIX publications. The procedures described in this manual use the following text conventions:

Table 1. General conventions

Convention	Meaning
Garamond blue type	Indicates a cross-reference hyperlink that points to a figure, graphic, table, or section heading. Clicking on the hyperlink jumps you to the reference. When you have finished reviewing the reference, click on the Go to Previous View button in the Adobe® Acrobat® Reader toolbar to return to your starting point.
Helvetica bold type	Commands and keywords are in boldface font.
Helvetica bold-italic type	Parts of commands, which are related to elements already named by the user, are in boldface italic font.
Italicized Helvetica type	Variables for which you supply values are in <i>italic</i> font
Garamond italic type	Indicates the names of fields or windows.
Garamond bold type	Indicates the names of command buttons that execute an action.
<>	Angle brackets indicate function and keyboard keys, such as <shift>, <ctrl>, <c>, and so on.
[]	Elements in square brackets are optional.
{a b c}	Alternative but required keywords are grouped in braces ({}) and are separated by vertical bars ()
<i>device</i>	The leading IP address or name of a Patton device is substituted with device in boldface italic font.
device	The leading device on a command line represents the name of the Patton device
#	An hash sign at the beginning of a line indicates a comment line.

Service and support

Patton Electronics offers a wide array of free technical services. If you have questions about any of our other products we recommend you begin your search for answers by using our technical knowledge base. Here, we have gathered together many of the more commonly asked questions and compiled them into a searchable database to help you quickly solve your problems.

Patton support headquarters in the USA

- Online support: Available at www.patton.com
- E-mail support: E-mail sent to support@patton.com will be answered within 1 business day
- Telephone support: Standard telephone support is available five days a week—from **8:00 am to 5:00 pm EST (1300 to 2200 UTC/GMT)**—by calling **+1 (301) 975-1007**
- Support via VoIP: Contact Patton free of charge by using a VoIP ISP phone to call [sip:support@patton.com](tel:sip:support@patton.com)
- Fax: **+1 (301) 869-9293**

Alternate Patton support for Europe, Middle East, and Africa (EMEA)

- Online support: Available at www.patton-inalp.com
- E-mail support: E-mail sent to support@patton-inalp.com will be answered within 1 business day
- Telephone support: Standard telephone support is available five days a week—from **8:00 am to 5:00 pm CET (0900 to 1800 UTC/GMT)**—by calling **+41 (0)31 985 25 55**
- Fax: **+41 (0)31 985 25 26**

Warranty Service and Returned Merchandise Authorizations (RMAs)

Patton Electronics is an ISO-9001 certified manufacturer and our products are carefully tested before shipment. All of our products are backed by a comprehensive warranty program.

Note If you purchased your equipment from a Patton Electronics reseller, ask your reseller how you should proceed with warranty service. It is often more convenient for you to work with your local reseller to obtain a replacement. Patton services our products no matter how you acquired them.

Warranty coverage

Our products are under warranty to be free from defects, and we will, at our option, repair or replace the product should it fail within one year from the first date of shipment. Our warranty is limited to defects in workmanship or materials, and does not cover customer damage, lightning or power surge damage, abuse, or unauthorized modification.

Returns for credit

Customer satisfaction is important to us, therefore any product may be returned with authorization within 30 days from the shipment date for a full credit of the purchase price. If you have ordered the wrong equipment or you are dissatisfied in any way, please contact us to request an RMA number to accept your return. Patton is not responsible for equipment returned without a Return Authorization.

Return for credit policy

- Less than 30 days: No Charge. Your credit will be issued upon receipt and inspection of the equipment.
- 30 to 60 days: We will add a 20% restocking charge (crediting your account with 80% of the purchase price).
- Over 60 days: Products will be accepted for repairs only.

RMA numbers

RMA numbers are required for all product returns. You can obtain an RMA by doing one of the following:

- Completing a request on the RMA Request page in the *Support* section at www.patton.com
- By calling +1 (301) 975-1007 and speaking to a Technical Support Engineer
- By sending an e-mail to returns@patton.com

All returned units must have the RMA number clearly visible on the outside of the shipping container. Please use the original packing material that the device came in or pack the unit securely to avoid damage during shipping.

Shipping instructions

The RMA number should be clearly visible on the address label. Our shipping address is as follows:

Patton Electronics Company
RMA#: xxxx
7622 Rickenbacker Dr.
Gaithersburg, MD 20879-4773 USA

Patton will ship the equipment back to you in the same manner you ship it to us. Patton will pay the return shipping costs.

Chapter 1 **System Overview**

Chapter contents

Introduction	52
Trinity embedded software	53
Applications	53
Carrier networks	54
Enterprise networks	54
LAN telephony	55

Introduction

This chapter provides an overview of the main elements of a Patton device system.

A complete Patton device system or network, as installed in any of the application scenarios introduced in section “Applications” on page 53, is typically composed of the following main elements plus a third-party network infrastructure:

- The first and most obvious element is the *Patton* devices (also referred to as *hardware platforms* or *network devices*) that provide the physical connectivity, the CPU and DSP resources. Some Patton device models support packet-routed and circuit-switched traffic, others do not.
- The second element comprises the embedded software—called *Trinity*—running on the Patton device hardware platforms.
- Finally, a third-party IP network and transmission infrastructure provides IP connectivity between the above elements. This infrastructure can range from a simple Ethernet hub or switch to highly complex networks including multiple access technologies, backbone transmission, and service devices.

Figure 1 depicts the basic system model of a Patton device. Patton VoIP + Router devices have the following main components:

- 64k circuit switching between on-board ISDN ports and between ISDN and PSTN interface cards. The circuit switching engine uses dedicated hardware resources and therefore can bypass the VoIP gateway and packet routing engine.
- A gateway (GW) that converts telephone circuits into Internet protocol (IP) packet streams and vice versa. SIP-compliant and SIP Voice over IP (VoIP) is supported.
- An IP router with on-board ports and optional data interface cards is QoS enabled, thereby allowing classification, shaping, and scheduling of multiple service classes.

Patton Router-only devices have the following main components:

- Physical data ports: Ethernet, DSL, Wifi, Cellular Modem, etc.
- Routing Core
- Firewall
- NAT

For more detailed hardware information, refer to the getting started guide that came with your Patton system.

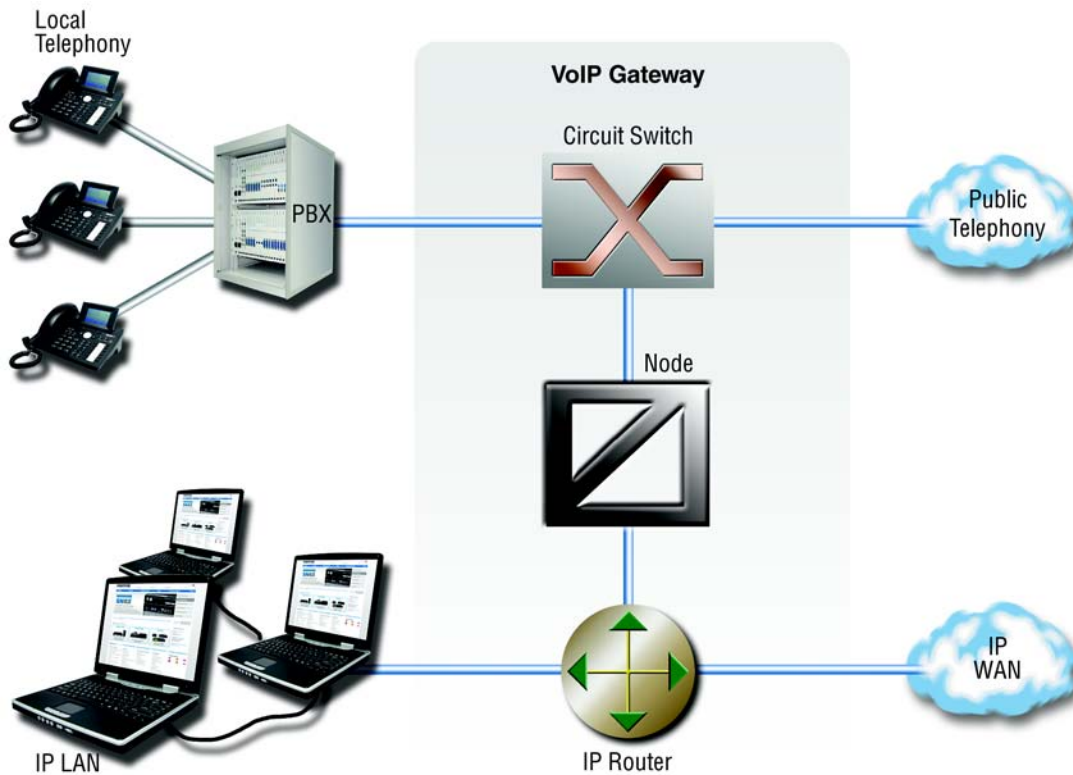


Figure 1. Basic system (abstract) model

Trinity embedded software

Trinity is the application software that runs on certain Patton device hardware platforms. Trinity is available in several releases.

A Trinity build is a binary image file. The download to the Patton device is packaged in a single TAR image that is uploaded onto the device. Refer to chapter 6, “[System Image Handling](#)” on page 84 for details on Trinity image downloads.

Applications

The Patton product family consists of highly flexible multi-service IP network devices, which fit a range of networking applications. This section provides an overview of the following Patton device applications and the main elements in a Patton network.

- Carrier networks—Patton devices are used as customer gateways or integrated access devices at the customer premises. These applications are also called Integrated Service Access (ISA).
- Enterprise networks—Patton devices are used as WAN routers and voice gateways for inter-site networking. These applications are also called *multiservice intranets* (MSI).
- LAN telephony—Patton devices serve as gateways between the LAN and the local PBX or PSTN access. These applications are also called LAN voice gateway (LVG).

Carrier networks

The network termination (NT) device in a multi-service IP based provider network plays a vital role. It provides the service access point for the subscriber with respect to physical connectivity and protocol interoperability.

Since the access bandwidth in most cases represents a network bottleneck, the NT must also ensure traffic classification and the enforcement of service level agreements (SLA) on the access link. In broadband access networks, this NT is also called an Integrated Access Device (IAD) or customer gateway.

Patton products offer unique features as customer gateways for business services. It provides amongst others full ISDN feature support, local switching and breakout options and mass provisioning support.

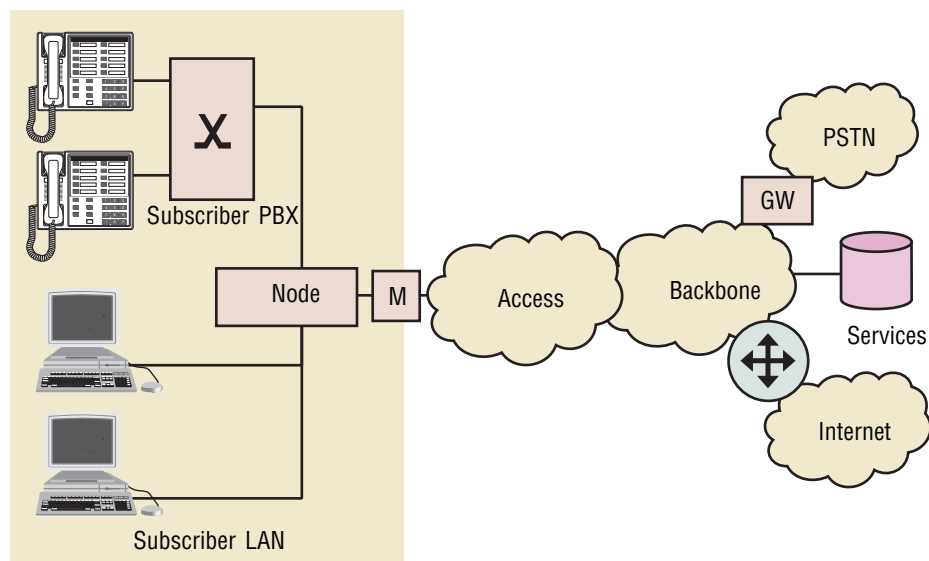


Figure 2. Typical carrier network application with a Patton device

Figure 2 shows the deployment of Patton devices in carrier networks. Each subscriber site is equipped with a Patton device that connects the subscriber LAN on one side with the provider network and services on the other.

Typical services in these networks are softswitch-based telephony, PSTN access through V5.2 gateways, PBX networking services, and LAN interconnection.

Typical access technologies for these networks include xDSL, WLL, PowerLine, cable and conventional leased lines. With the use of an external modem, the device can connect to leased lines or any bridged-Ethernet broadband access.

Enterprise networks

In company-owned and operated wide area networks, Patton devices can be used to converge voice and data communications on the same IP link. In combination with centralized services such as groupware and unified messaging, Patton devices provide migration and investment protection for legacy telephony systems.

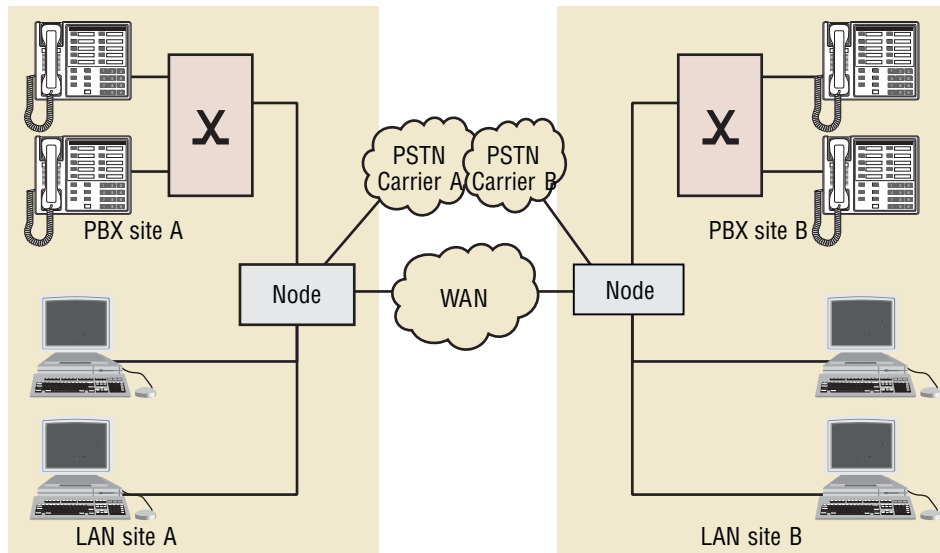


Figure 3. Typical enterprise network with a Patton device

Figure 3 shows the deployment of Patton devices in enterprise networks. Each site (headquarter, branch or home office) is equipped with a Patton device that connects the local LAN and telephony infrastructure with the IP WAN and the local PSTN carrier.

LAN telephony

With its voice-over-IP gateway features, the Patton device can be used as a standalone gateway for VoIP telephony (see figure 4).

A standalone gateway has performance reliability and scalability advantages compared with PC-based gateway cards. In this application, the Patton device also offers a migration path to enterprise or carrier networking.

Figure 4 shows the deployment of a Patton device as a LAN voice gateway.

The PSTN connections can be scaled from a single ISDN basic rate access to multiple primary rate lines. With Q.SIG, integration in private PBX networks is also supported.

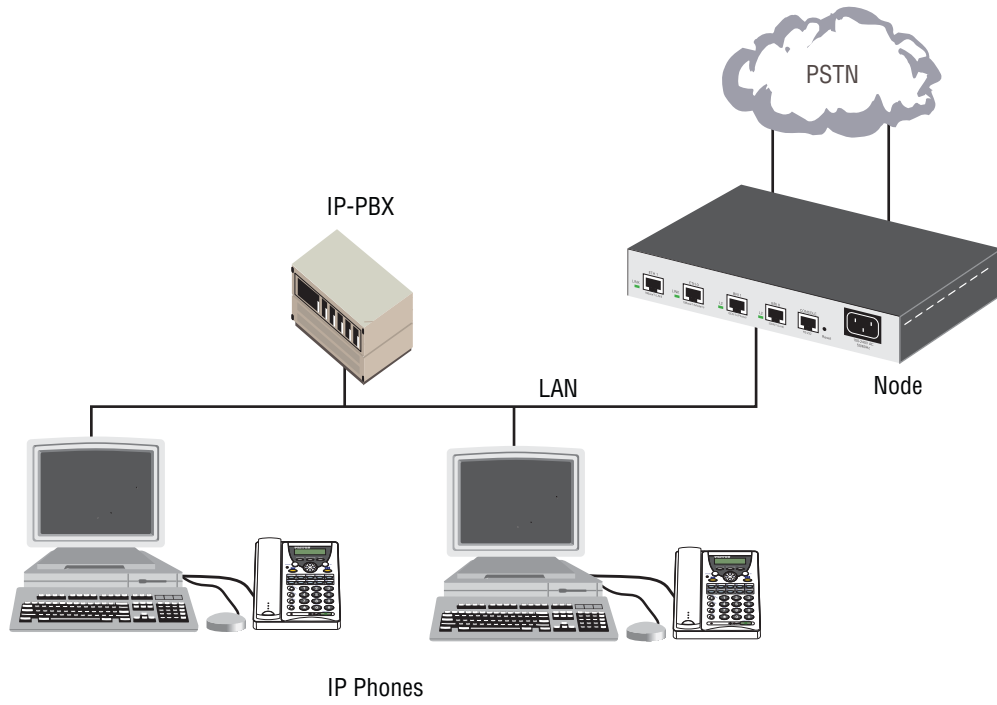


Figure 4. Typical LAN telephony system with a Patton device gateway

Chapter 2 **Configuration Concepts**

Chapter contents

Introduction.....	58
Contexts and Gateways.....	59
Context	59
Gateway	59
Interfaces, Ports, and Bindings	60
Interfaces	60
Ports and circuits	60
Bindings	60
Profiles and Use commands.....	61
Profiles	61
Use Commands	61

Introduction

This chapter introduces basic Trinity configuration concepts. A good understanding of these concepts is vital for the configuration tasks explained in the remaining chapters of this guide.

Patton strongly recommends that you read through this chapter because it introduces the fundamental ideas behind the structure of the command line interface. Once you understand and know this structure, you will find it much more intuitive to navigate through the CLI and configure specific features.

This chapter includes the following sections:

- Contexts and gateways (see [page 59](#))
- Interfaces, ports, and bindings (see [page 60](#))
- Profiles and Use commands (see [page 61](#))

Patton devices are multi-service network devices that offer high flexibility for the inter-working of circuit-switched and packet-routed networks and services. In order to consistently support a growing set of functions, protocols, and applications, Trinity configuration is based on a number of abstract concepts that represent the various Trinity components.

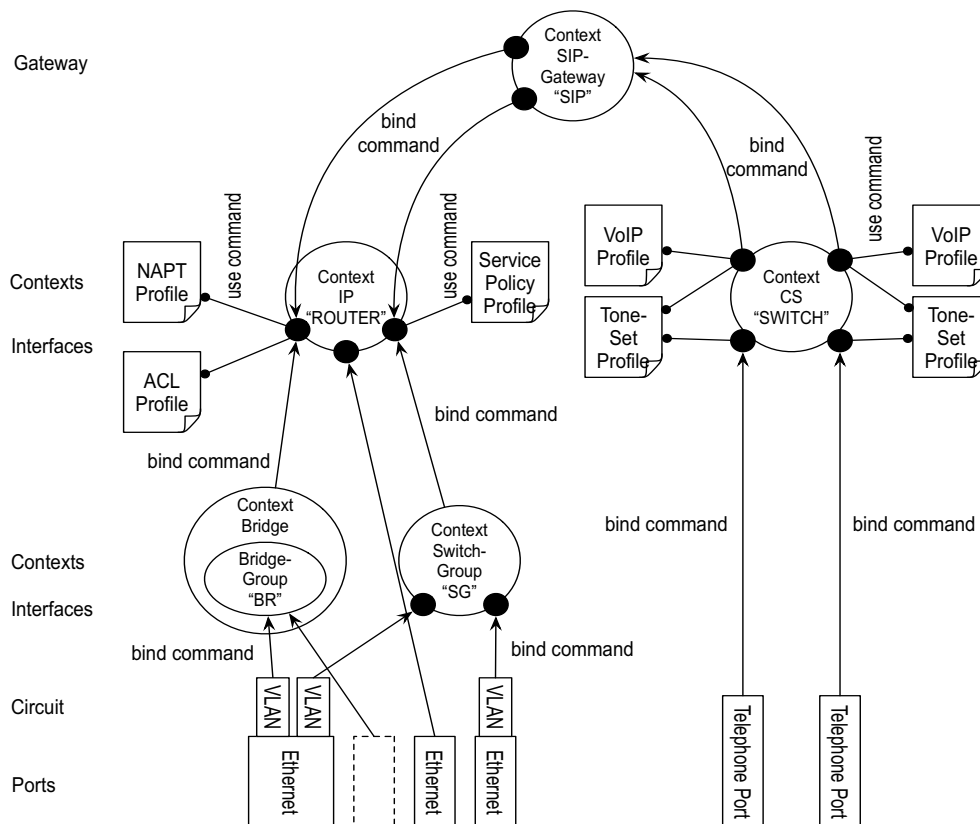


Figure 5. Configuration concept overview

Figure 5 shows the various elements of a complete Patton device configuration. Each of these elements implements one of the configuration concepts described in this chapter. The figure also shows the relationships and

associations between the different elements. The relations are specified through *bind* (arrow) and *use* (bullet-lines) commands. For example, you need *bind* commands to bind a physical port to a logical interface, and *use* commands to assign profiles to contexts.

The sections that follow refer to figure 5 on page 58 and describe the concepts and elements in more detail.

Contexts and Gateways

Context

A *context* represents one specific networking technology or protocol, namely IP (Internet Protocol) or CS (circuit-switching). A context can be seen as *virtual dedicated equipment* within the Patton device. For example:

- A CS context contains the circuit-switching functions of the Patton device. It can be thought of as an embedded multiplexer or cross-connect within the device
- An IP context contains the routing functions of the Patton device. It can be thought of as an embedded router within the device
- A Bridge context contains the bridging functions of the Patton device at the CPU layer. Software bridging and bridge management is configured within.
- A Switch-group context contains the bridging functions of the Patton device at the hardware layer.

The contexts are identified by a name and contain the configuration commands that are related to the technology they represent. A separate configuration can be built by means of the context concept for newly supported network layer technologies without complicating the configuration methods of existing features. For example, as ATM or FR switching becomes available so an ATM or FR context can be introduced.

Each context contains a number of *interfaces*, which build the connections to other Trinity elements and the outside world. Figure 5 on page 58 shows four contexts:

- one type IP named *router*
- one type CS named *switch*
- one type Bridging
- one type of Switch-group

Note Trinity currently supports only one instance of the CS and IP context types.

Example

The IP context named *router* can contain static routes, RIP, and NAT configuration parameters. The default circuit-switching context named *switch* can contain number translations, local breakout conditions, and least-cost routing parameters.

Gateway

The concept of a *gateway* is introduced for the communication between contexts of different types. A gateway handles connections between different technologies or protocols. For example, a VoIP gateway connects an IP context to a circuit-switching context.

The gateways are each of a specific type and are identified by a name. Each named gateway contains its configuration parameters. With this concept, multiple virtual gateways can be instantiated and used at the same time.

Interfaces, Ports, and Bindings

Interfaces

The concept of an interface in Trinity differs from that in traditional networking devices. Traditionally, the term *interface* is often synonymous with *port* or *circuit*, which are physical entities. In Trinity however, an interface is a logical construct that provides higher-layer protocol and service information, such as layer 3 addressing. Interfaces are configured as part of a context, and are independent of physical ports and circuits. The decoupling of the interface from the physical layer entities enables many of the advanced features offered by Trinity.

In order for the higher-layer protocols to become active, you must associate an interface with a physical port or circuit. This association is referred to as a *binding* in Trinity. Refer to the “[Bindings](#)” section for more information. In figure 5 on page 58, the IP context shows three interfaces and the CS context shows four interfaces. These interfaces are configured within their contexts. The bindings shown in the figure are not present when the interfaces are configured; they are configured later.

Ports and circuits

Ports and *circuits* in Trinity represent the physical connectors and channels on the Patton device hardware. The configuration of a port or circuit includes parameters for the physical and data link layer such as line clocking, line code, framing and encapsulation formats or media access control. Before any higher-layer user data can flow through a physical port or circuit, you must associate that port or circuit with an interface on a context. This association is referred to as a *binding*. Refer to the “[Bindings](#)” section for more information.

Examples of ports are: DSL, BRI, E1/T1, or FXS. Ports are numbered according to the label (or abbreviation) printed on the hardware.

Example: Ethernet 0/1, BRI 3/2

Figure 5 on page 58 shows five ports. Three ports are bound directly to an IP interface. One port has a single circuit configured, which is bound to the IP context. Two ISDN ports are bound to CS interfaces.

Bindings

Bindings form the association between circuits or ports and the interfaces configured on a context. No user data can flow on a circuit or Ethernet port until some higher-layer service is configured and associated with it.

Bindings are configured statically in the port or circuit configuration. The binding is created bottom-up, that is from the port to the interface.

In the case of VoIP CS interfaces, bindings are configured statically in the CS interface configuration. The binding is created from the interface to the gateway.

Bindings from ports to interfaces shown in figure 5 on page 58.

Profiles and Use commands

Profiles

Profiles provide configuration shortcuts. They contain specific settings that can be used in multiple contexts, interfaces, or gateways. This concept allows to avoid repetitions of groups of configuration commands that are the same for multiple elements in a configuration.

Profiles used in the IP and CS contexts are shown in figure 5 on page 58.

Use Commands

Use commands form the association between profiles and contexts, gateways, or interfaces. For example, when a profile is *used* in a context, all the configuration settings in that profile become active within the context.

Chapter 3 **Command Line Interface (CLI)**

Chapter contents

Introduction	63
Command modes	63
CLI prompt	63
Navigating the CLI	63
Initial mode	63
System changes	64
Configuration	64
Changing Modes	64
Command editing	64
Command help	64
The No Form	64
Command completion	64
Command history	65
Command Editing Shortcuts	65
Timed Execution of CLI Command	65

Introduction

The primary user interface to Trinity is the command line interface (CLI). You can access the CLI via the Patton device console port or through a Telnet or SSH session. The CLI lets you configure the complete Trinity functionality. You can enter CLI commands online or as a configuration script in the form of a text file. The CLI also includes monitoring and debugging commands. CLI commands are simple strings of keywords and user-specified arguments.

This chapter gives an overview of the CLI and the basic features that allow you to navigate the CLI and edit commands effectively. The following topics are covered:

- Command Modes
- Command Editing (see [page 64](#))

Command modes

The CLI is composed of modes. There are three *mode groups*: the operator, the *administrator mode* and the *configure mode*. The configuration mode group contains all of the remaining modes. A command mode is an environment within which a group of related commands is valid. All commands are mode-specific, and certain commands are valid in more than one mode. A command mode provides command line completion and context help within the mode. The command modes are organized hierarchically. The current working mode is indicated by the CLI prompt.

CLI prompt

For interactive (online) sessions, the system prompt is displayed as:

```
devicename>
```

In the operator exec mode, the system prompt is displayed as:

```
devicename#
```

In the administrator exec mode and in the different configuration modes, the system prompt is displayed as:

```
devicename(mode)device#
```

Where:

- *devicename* is the currently configured name of the Patton device, the IP address or the hardware type of the device that is being configured
- *mode* is a string indicating the current configuration mode, if applicable.
- *name* is the name of the instance of the current configuration mode

Navigating the CLI

Initial mode

When you initiate a session, you can log in with operator or administrator privileges. Whichever login you use, the CLI is always set to operator exec (non-privileged exec) mode by default upon startup. This mode allows you to examine the state of the system using a subset of the available CLI commands.

System changes

In order to make changes to the system, the administrator `exec` (privileged `exec`) mode must be entered. The **enable** user interface command is used for this purpose (the **enable** command is only accessible if you are logged in as an administrator). Once in administrator `exec` mode, all of the system commands are available to you.

Configuration

To make configuration changes, the configuration mode must be entered by using the **configure** command in the administrator `exec` mode.

Changing Modes

The **exit** command moves the user up one level in the mode hierarchy (the same command works in any of configuration modes).

The **exit** command terminates a CLI session when typed from the operator `exec` mode.

A session can also be terminated by using the **logout** command within any mode.

Command editing

Command help

To see a list of all CLI commands available within a mode, type a question mark `<?>` or the `<tab>` key at the system prompt in the mode of interest. A list of all available commands is displayed. Commands that have become available in the current mode are displayed at the bottom of the list, separated by a line. Commands from higher hierarchy levels are listed at the top.

You can also type the question mark or the `<tab>` key while in the middle of entering a command. Doing so displays the list of allowed choices for the current keyword in the command. Liberal use of the question mark functionality is an easy and effective way to explore the command syntax.

The No Form

Almost every command supports the keyword **no**. Typing the **no** keyword in front of a command disables the function or “deletes” a command from the configuration. For example, to enable the DHCP server trace tool, enter the command **debug dhcp-server**. To subsequently disable the DHCP server trace, enter the command **no debug dhcp-server**.

Command completion

You can use the `<tab>` key in any mode to carry out command completion. Partially typing a command name and pressing the `<tab>` key causes the command to be displayed in full up to the point where a further choice has to be made. For example, rather than typing **configure**, typing **conf** and pressing the `<tab>` key causes the CLI to complete the command at the prompt. If the number of characters is not sufficient to uniquely identify the command, the CLI will provide a list with all commands starting with the typed characters. For example, if you enter the string `co` in the configure mode and press `<tab>`, the selections **configure**, **copy**, and **context** are

displayed. The CLI may be configured to automatically complete commands without pressing the <tab> key. This will only happen if a unique completion option exists.

Command	Purpose
[no] cli auto-completion	Enable or disable CLI automatic command completion.

Command history

Trinity maintains a list of previously entered commands that you can go through by pressing the <up-arrow> and <down-arrow> keys, and then pressing <enter> to enter the command. The show history command displays a list of the commands you can go through by using the arrow keys.

Command Editing Shortcuts

Trinity CLI provides a number of command shortcuts that facilitate editing of the command line. Command editing shortcuts are summarized below. The syntax <Ctrl>-<p> means press the <p> key while holding down the keyboard's control key (sometimes labeled *Control*, *Ctl*, or *Ctrl*, depending on the keyboard and operating system of your computer). <Esc>-<f> is handled differently; press and release the escape key (often labeled *Esc* on many keyboards) and then press the <f> key.

Keyboard	Description
<Ctrl>-<p> or <up-arrow>	Recall previous command in the command history.
<Ctrl>-<n> or <down-arrow>	Recall next command in the command history.
<right-arrow>	Move cursor forward one character.
<left-arrow>	Move cursor backward one character.
<Esc>-<f>	Move cursor forward one word.
<Esc>-	Move cursor backward one word.
<Ctrl>-<a>	Move cursor to beginning of line.
<Ctrl>-<e>	Move cursor to end of line.
<Ctrl>-<k>	Delete to end of line.
<Ctrl>-<u>	Delete to beginning of line.
<Ctrl>-<d>	Delete character.
<Ctrl>-<c>	Quit editing the current line.
<Ctrl>-<v>	Insert a code to indicate to the system that the keystroke immediately following should be treated as normal text, not a CLI command. For example, pressing the question mark <?> character in the CLI prints a list of possible tokens. If you want to use the "?" in a configuration command, e.g. to enter a regular expression, press Ctrl-v immediately followed by the question mark <?>.

Timed Execution of CLI Command

The command timer allows the timed execution of CLI commands. The timer command is incremental; this means for each time it is entered, a new timer is created. All timers appear in the running-configuration, except

if they have been created with the volatile option. It is possible to specify for each timer the start time and the reoccurrence. Use the CLI help (tab completion) for detailed description of all configuration options.

Example:

```
timer FIRMWARE_UPDATE now + 2 minutes every 10 minutes "provisioning execute FIRM-WARE"
```

Starts a timer named FIRMWARE_UPDATE, whose first execution time is 2 minutes after the command is entered (2 minutes after device startup if the command is in the startup-configuration), and is executed every 10 minutes afterwards. This timer does not expire. The executed CLI command is provisioning execute FIRMWARE.

As there are many possibilities to configure the timer time specification, here are some practical examples:

```
timer MYTIMER every day "command"
```

Will execute the command "command" every day at the same time.

```
timer MYTIMER oct 9th 2019 "command"
```

Will execute the command "command" on October 9th 2019 at midnight.

```
timer MYTIMER now + 2 minutes every month "command"
```

Will execute the command "command" in 2 minutes and then every month at the same hour.

Mode: configure

Step	Command	Purpose
1	node (cfg)#timer (volatile)<name> <time specification>	Enter the timer

Chapter 4 **Accessing the CLI**

Chapter contents

Introduction	68
Accessing the Trinity CLI task list	68
Accessing via the console port	69
Console port procedure	69
Accessing via a secure configuration session over SSH	69
Accessing via a Telnet session	70
Telnet Procedure	70
Using an alternate TCP listening port for the Telnet or SSH server	70
Disabling the Telnet or SSH server	70
Logging on	70
Selecting a secure password	71
Password encryption	72
Factory preset superuser account	72
Configuring operators, administrators, and superusers	72
Creating an operator account	72
Creating an administrator account	73
Creating a superuser account	74
Displaying the CLI version	75
Displaying account information	75
Checking identity and connected users	75
Command index numbers	76
Ending a Telnet, SSH or console port session	78

Introduction

Patton products are designed for remote management and volume deployment. The management and configuration of Patton devices is therefore based on IP network connectivity. Once a Patton device is connected to, and addressable in, an IP network, you can remotely perform all configuration, management, and maintenance tasks.

This chapter describes the procedures for entering Trinity commands via the command line interface (CLI), to obtain help, to change operator mode, and to terminate a session. You can access a Patton device as follows:

- Directly, via the console port (if available)
- Remotely, via the IP network (by using a Telnet or SSH application)

The ports available for connection and their labels are shown in the getting started guide that came with your unit. Remember that the CLI supports a command history and command completion. By scrolling with the **up** and **down** arrow keys, you can find many of your previously entered commands. Another time-saving tool is command completion. If you type part of a command and then press the **<tab>** key, the Trinity shell will present you with either the remaining portion of the command or a list of possible commands. These features are described in Chapter 3, “[Command Line Interface \(CLI\)](#)” on page 62. The telnet and SSH server can be disabled if desired.



Although Trinity supports concurrent sessions via SSH or the console port, we do not recommend working with more than one session to configure a specific Patton device. However, using one session for configuration and another for debugging is a good idea.

Accessing the Trinity CLI task list

The following sections describe the basic tasks involved in accessing the Trinity command line interface. Depending on your application scenario, some tasks are mandatory while others could be optional.

- Accessing via the console port (see [page 69](#))
- Accessing via a SSH session (see [page 70](#))
- Using an alternate TCP listening port for the SSH server (see [page 70](#))
- Disabling the SSH server (see [page 70](#))
- Logging on (see [page 70](#))
- Selecting a secure password (see [page 71](#))
- Configuring operators and administrators (see [page 72](#))
- Displaying the CLI version (see [page 75](#))
- Switching to another log-in account (see [page 75](#))
- Checking identity and connected users (see [page 75](#))
- Ending a SSH or console port session (see [page 78](#))

Accessing via the console port

If a console port is available, the host computer can be connected directly to it with a serial cable (see Figure 6). The host must use a terminal emulation application that supports serial interface communication.



Figure 6. Setup for initial configuration via the console port

Note You do not need to configure IP settings if you access the Patton device via the console port.

Console port procedure

Before using the CLI to enter configuration commands, do the following:

1. Set up the hardware as described in the getting started guide.
2. Configure your serial terminal as described in the getting started guide.
3. Connect the serial terminal to your Patton device. Use a serial cable according to the description in the getting started guide included with your Patton device.
4. Power on your device. A series of boot messages are displayed on the terminal screen. At the end of the boot sequence, press the **<return>** key and the login screen will be displayed. Proceed with logging in.

Accessing via a secure configuration session over SSH

SSH is the most commonly used and recommended method for connecting to a Patton device. A partial implementation of secure shell according RFC 4251, RFC 4252, RFC 4253 and RFC 4254 is provided. It is possible to open a secure configuration session over SSH to a Patton device.

Note The copy tftp and http functions are still insecure!

The SSH Transport Layer supports the following Algorithms: “ssh-rsa” or “ssh-dsa” public key for signing, “diffie-hellmann-group1-sha1” and “diffie-hellmann-group14-sha1” for key exchange, “3des-cbc”, “aes256-cbc” and “aes128-cbc” for encryption, “hmac-sha1” and “hmac-md5” for data integrity. For user authentication, only the method “password” is supported. On the Connection Layer, only the request for an interactive command shell is supported. After the first startup of Trinity, the RSA or DSA server host key is going to be calculated. The RSA or DSA server host key is calculated only once and always remains the same.

Mode: Configure

Step	Command	Purpose
1	device(cfg)#terminal ssh use auth <AAA profile name>	Set the AAA profile which is going to be used for user authentication. The AAA profile “default” is used when another profile is not specified.

Accessing via a Telnet session

It is way faster than console access. The Telnet host accesses the Patton device via its network interface.

Note If the IP configuration of the Ethernet port (LAN port) is not known or is incorrectly configured, you will have to use the console interface.

Telnet Procedure

Before you begin to use the CLI to input configuration commands, do the following:

1. Set up the Patton device as described in the getting started guide included with your device.
2. Connect the host (PC) or hub to the Patton device as described in the getting started guide.
3. Power on your device and wait until the *Run* LED lights.
4. Open a Telnet session to the IP address shown in the getting started guide.
5. Proceed with logging in.

Using an alternate TCP listening port for the Telnet or SSH server

The following command defines an alternate listening port for the telnet or SSH server.

Mode: Configure

Step	Command	Purpose
1	telnet-server port <port> or ssh-server port <port>	Uses TCP port <port> for accepting telnet or SSH connections

Disabling the Telnet or SSH server

The telnet or SSH server can be disabled using the following command.

Mode: Configure

Step	Command	Purpose
1	device(cfg)# no terminal [telnet ssh]	Disables the telnet or SSH server

Logging on

Accessing your Patton device via the local console port or via a Telnet session opens a login screen. The following description of the login process is based on a Telnet session scenario but is identical to that used when accessing via the local console port.

The opening Telnet screen you see resembles that shown below. The window header bar shows the IP address of the target Patton device.

A factory preset superuser account with name *admin* and an empty password is available when you first access the unit. For that reason, use the name *admin* after the login prompt and simply press the **<enter>** key after the password prompt.

```
$ telnet 172.16.54.79
Trying 172.16.54.79...
```

```
Connected to 172.16.54.79.  
Escape character is '^]'.  
  
Patton Electronics Company FF3310RC  
Release: 3.1.0 2013/01/20  
  
Trinity login: admin  
Password:  
  
Trinity >
```

Upon logging in you are in operator execution mode, indicated by the “>” as command line prompt. Now you can enter system commands.

Note Details on the screen, such as the IP address in the system prompt and window header bar, may be different on your unit.



You are responsible for creating a new administrator account to maintain system security. Patton Electronics accepts no responsibility for losses or damage caused by loss or misuse of passwords. Please read the following sections to secure your network equipment properly.

Selecting a secure password

It is not uncommon for someone to try to break into (often referred to as *hacking*) a network device. The network administrator should do everything possible to make the network secure. Carefully read the questions below and see if any applies to you:

- Do your passwords consist of a pet’s name, birthdays or names of friends or family members, your license plate number, social security number, favorite number, color, flower, animal, and so on?
- Do you use the same password repeatedly? (Example: Your ATM PIN, cell phone voice mail, house alarm setting code, etc.)
- Could your password or a portion thereof be found in the dictionary?
- Is your password less than six characters long?

To prevent unauthorized access, you should select passwords that are not dictionary words or any of the above-mentioned examples. Every password should be at least 6 characters long and include at least one capital letter, one number, and one lowercase letter.

A good example of a password is: *3Bmsbtr*

You are probably asking yourself, “How am I going to remember that?” It’s easy, the password above is an acronym taken from: “three blind mice, see how they run.” Making a good password is that easy—but please, don’t use the above example password for your Patton device!

Password encryption

Unencrypted passwords can be stolen by hackers using protocol analyzers to scan packets or by examining the configuration file—to protect against that type of theft, Trinity encrypts passwords by default. Encryption prevents the password from being readable in the configuration file.

- Plain text
- Encrypted text (for example, the password mypassword always appears in encrypted form as *HUAvCYeILW-Zz3hQvSOIEpQ==* encrypted when doing a **show** command)

The command **show running-config** always displays the passwords in encrypted format. To encrypt a password, enter the password in plain format and retrieve the encrypted format from the running-config or store it permanently into the startup-config (with the command **copy running-config startup-config**).

Factory preset superuser account

Trinity contains a factory preset superuser account with the name *admin* (no passwords). When a new superuser account has been defined in the configuration, the preset admin account will delete after reboot. You can create more than one superuser account, but there has to be at least one superuser account defined. If, for some reason, the last superuser account is deleted, the factory preset administration account with the name *admin* and an empty password is automatically recreated.

Configuring operators, administrators, and superusers

Creating an operator account

The operator can only show a small set of states for supervising the functionality of a device. He does not configure or change anything on the device.

- **Default not allowed:**
 - Config write commands
 - Exec commands
 - Copy config commands
 - Software upgrade
- **Default allowed:**
 - Debug/trace commands
 - Show commands (not all)
 - Show running config (incomplete)
- **Exceptions to default behavior:** show accounts (and related) is not allowed.

Creating a new operator account is described in the following procedure:

Mode: Operator execution

Step	Command	Purpose
1	<i>device</i> >enable	Enters administration execution mode

Step	Command	Purpose
2	<code>device#configure</code>	Enters configuration mode
3	<code>device(cfg)# operator name password password</code>	Creates a new operator account <i>name</i> and password <i>password</i>
4	<code>copy running-config startup-config</code>	Saves the change made to the running configuration of the Patton device, so that it will be used following a reload

Example: Create an operator account

The following example shows how to add a new operator account with a login name *support* and a matching password of *s4DF&qw*. The changed configuration is then saved.

```
device>enable
device#configure
device(cfg)#operator support password s4DF&qw
device(cfg)#copy running-config startup-config
```

Creating an administrator account

The administrator can configure and debug a device in operation. The main exception is he is not allowed to configure user accounts. Therefore all configuration which allows to remove accounts from a device in an easy way are not allowed. Remark: We cannot prevent a physical present administrator/operator to gain supervisor access to a device via erasing the config from uboot.

- **Default not allowed:**
 - Any related to Provisioning
 - Any related to AAA
 - Any related to User accounts
- **Default allowed:**
 - Software upgrade
 - Copy config commands
 - Config write commands
 - Exec commands
 - Debug/trace commands
 - Show commands
 - Show running config (complete)

Creating a new administrator account is described in the following procedure:

Mode: Operator execution

Step	Command	Purpose
1	<code>device>enable</code>	Enters administration execution mode
2	<code>device#configure</code>	Enters configuration mode

Step	Command	Purpose
3	<code>device(cfg)# administrator <i>name</i> password <i>password</i></code>	Creates a new administrator account <i>name</i> and password <i>password</i>
4	<code>device(cfg)#copy running-config startup-config</code>	Permanently stores the new administrator account parameters.

Example: Create an administrator account

The following example shows how to add a new administrator account with a login name *super* and a matching password *Gh3*Ke4h*.

```
device>enable
device#configure
device(cfg)#administrator super password Gh3*Ke4h
device(cfg)#copy running-config startup-config
```

A web wizard account is a user that can only configure the system using Web Wizards. It cannot do software upgrades nor manage wizards files or configuration files. A Web Wizard account can be created via the CLI as follows:

Mode: Configure

Step	Command	Purpose
1	<code>device(cfg)#administrator <username> password <user-password> terminal-type web web-wizard-exec</code>	Add a new user who can only access and configure the device via the Web Wizards.

Creating a superuser account

The superuser is the main administrator which can configure the device and manages user accounts.

- **Default allowed:** Everything
- **Exceptions to default behavior:** engineer exec commands are not allowed

Creating a new superuser account is described in the following procedure:

Mode: Operator execution

Step	Command	Purpose
1	<code>device>enable</code>	Enters administration execution mode
2	<code>device#configure</code>	Enters configuration mode
3	<code>device(cfg)# superuser <i>name</i> password <i>password</i></code>	Creates a new superuser account <i>name</i> and password <i>password</i>
4	<code>device(cfg)#copy running-config startup-config</code>	Permanently stores the new superuser account parameters.

Example: Create a superuser account

The following example shows how to add a new superuser account with a login name *super* and a matching password *Gh3*Ke4h*.

```
device>enable
device#configure
device(cfg)#superuser super password Gh3*Ke4h
device(cfg)#copy running-config startup-config
```

Displaying the CLI version

This procedure displays the version of the currently running CLI.

Mode: Operator execution

Step	Command	Purpose
1	<code>device>show version cli</code>	Displays the CLI version

Example: Displaying the CLI version

The following example shows how to display the version of the current running CLI on your device, if you start from the operator execution mode.

```
device>show version cli
CLI version: 3.00
```

Displaying account information

You can use the **show** command to display information about existing administrator and operator accounts. This command is not available for an operator account.

The following procedure describes how to display account information:

Mode: Administrator execution

Step	Command	Purpose
1	<code>device#show accounts</code>	Displays the currently-configured administrator and operator accounts.

Example: Display account information

The following example shows how to display information about existing administrator and operator accounts.

```
device#show accounts
# UserName AccessLevel Status
0 super superuser (logged out:0)
1 admin administrator (logged out:0)
2 op operator (logged out:0)
```

Checking identity and connected users

The **who** command displays who is logged in or gives more detailed information about users. Depending on the execution mode, the command displays varying information. In administrator execution mode, the command output is more detailed and shows information about the ID, user name and location. In operator execution mode, only the user name being used at the moment is reported, which helps checking the identity.

Mode: Administrator or operator execution

Step	Command	Purpose
1	<i>Trinity(cfg)#who</i>	Shows more detailed information about the users ID, name, state, idle time and location
or		
	<i>Trinity>who</i>	Shows the user login identity

Example: Checking identity and connected users

The following example shows how to report who is logged in or more detailed information about users, depending on the execution mode in which you are working.

Used in administrator execution mode:

```
Trinity(cfg)#who
# User Name          Login Time          Location
0 admin             01/01/2000 00:08:59 console
1 admin             01/01/2000 00:11:36 telnet 172.16.54.135:55404
```

Used in operator execution mode:

```
Trinity>who
You are operator support
```

Command index numbers

A command index number (indicated by the boldface **1**, **2**, and **3** index numbers in the example below) indicates the position of a command in a list of commands (that is, a command with index *1* will appear higher in the configuration file than one with index *3*).

```
192.168.1.1(pf-voip)[default]#show running-config
...
profile voip DEFAULT
codec 1 g711ulaw64k rx-length 20 tx-length 20
codec 2 g711alaw64k rx-length 20 tx-length 20
codec 3 g723-6k3 rx-length 30 tx-length 30
de jitter-max-delay 200
...
```

Commands that make use of index numbers always show the index in the running config. However, the index can be omitted when entering the command. If you enter such a command with an index, it is inserted into list at the position defined by the index. If you enter such a command without an index, it is placed at the bottom of the list. Also, you can change a commands position in a listing (moving it up or down in the list) by changing its index number.

Example 1: Moving the G.723 codec from position *3* in the list to position *1* at the top of the list.

Listing before changing the G.723 codec index number:

```
profile voip DEFAULT
codec 1 g711ulaw64k rx-length 20 tx-length 20
codec 2 g711alaw64k rx-length 20 tx-length 20
codec 3 g723-6k3 rx-length 30 tx-length 30
de jitter-max-delay 200
```


...

Listing after changing index number:

```
192.168.1.1(pf-voip)[default]#codec 3 before 1
192.168.1.1(pf-voip)[default]#show running-config
...
profile voip DEFAULT
codec 1 g723-6k3 rx-length 30 tx-length 30
codec 2 g711ulaw64k rx-length 20 tx-length 20
codec 3 g711alaw64k rx-length 20 tx-length 20
dejitter-max-delay 200
...
```

Note Succeeding indexes are automatically renumbered.

Example 2: Moving the G.723 codec back position 3

This command moves the G.723 codec from the top to third place. As a result, the other two codecs move up in the list as their indexes are automatically renumbered to accommodate the new third-place codec.

```
192.168.1.1(pf-voip)[default]#codec 1 after 3
192.168.1.1(pf-voip)[default]#show running-config
...
profile voip DEFAULT
codec 1 g711ulaw64k rx-length 20 tx-length 20
codec 2 g711alaw64k rx-length 20 tx-length 20
codec 3 g723-6k3 rx-length 30 tx-length 30
dejitter-max-delay 200
...
```

Example 3: Inserting a codec at a specific position in the list.

This command assigns the G.729 codec the index number 1 so the codec appears at the top of the list.

```
192.168.1.1(pf-voip)[default]#codec 1 g729 tx-length 30 rx-length 30 silence-sup-
pression
192.168.1.1(pf-voip)[default]#show running-config
...
profile voip DEFAULT
codec 1 g729 rx-length 30 tx-length 30 silence-suppression
codec 2 g711ulaw64k rx-length 20 tx-length 20
codec 3 g711alaw64k rx-length 20 tx-length 20
  codec 4 g723-6k3 rx-length 30 tx-length 30
dejitter-max-delay 200
...
```

Ending a Telnet, SSH or console port session

Use the **logout** command in the operator or administration execution mode to end a Telnet or console port session. To confirm the **logout** command, you must enter **yes** on the dialog line as shown in the example below.

Mode: Operator execution

Step	Command	Purpose
1	<i>device></i> logout	Terminates the session after a confirmation by the user.

Example: End a Telnet or console port session

The following example shows how to terminate a session from the administrator execution configuration mode.

```
device>logout
Press 'yes' to logout, 'no' to cancel:
```

After confirming the dialog with “yes”, the Telnet session is terminated.

Note Using the command **exit** in the operator execution mode also terminates a Telnet or console port session, but without any confirmation dialog.

Step	Command	Purpose
1	<i>device></i> enable	Enters administration execution mode
2	<i>device#</i> configure	Enters configuration mode
3	<i>device(cfg)#</i> superuser <i>name</i> password <i>password</i>	Creates a new superuser account <i>name</i> and password <i>password</i>
4	<i>device(cfg)#</i> cli config defaults	Generate a command even if it reflects the default setting (Default: Disabled)

Chapter 5 **Creating CLI Action Scripts**

Chapter contents

Introduction	80
Action Script Task List	80
Creating an Action Script	80
Conditions	80
Context CS Events	81
SIP Interface Events	81
Context IP Events	81
SIP Gateway Events	82
System NTP Events	82
System Timer Events	82
Actions	83

Introduction

Trinity's event-driven user-programmed hook system allows users to create a specific CLI script to execute on a specific event (for example, link down on the IP interface).

Action Script Task List

To configure Action Scripts, perform the tasks in the following sections:

- Creating an Action Script
- Conditions
 - Events
- Actions

Creating an Action Script

There are several parameters that need to be configured in order for the Action Script to properly function. Those steps are listed below.

Mode: configure

Step	Command	Purpose
1	device(cfg)#[no] actions	Enters the actions configurations
2	device(act)[<rule-name>]#[no] rule <name>	Creates/deletes a rule

Conditions

There can be more than one condition per rule. The rule will be executed each time one of the conditions match the event. This gives the possibility to trigger an Action Script by several events. All events are deferred until the startup-config is parsed to avoid any error due to partial configuration parsing.

Mode: actions

Step	Command	Purpose
1	device(act)[<rule-name>]#[no] condition <group-name> <source-name> <event-name> [initial]	Add or remove a condition to the rule. The rule is executed as soon as one of the conditions matched an event. The <i>initial</i> option means that the condition will only be matched by the first event (until next reboot). This can be used for if you want to only match the first link up event after boot up.

Context CS Events

Triggers an action script based on ISDN Interface state change, i.e when ISDN interface is up or down.

Step	Command	Purpose
1	device(act)[<rule-name>]#[no] condition cs isdn:<if-name> {LINKUP LINKDOWN} [initial]	Add or remove a condition to the rule based on events within the context cs ISDN interfaces.

SIP Interface Events

Mode: actions

Triggers an action script based on the SIP Interface's reachability of SIP remotes.

Step	Command	Purpose
1	node(act)# rule <name>	Creates/Enters an actions script rule.
2	node(act)[<name>]# condition cs sip:<if-name> { OPTIONS_TARGET_NOT_REACHABLE OPTIONS_TARGET_REACHABLE }	Enter the condition in which the following action(s) have to be executed.
3	node(act)[<name>]# action <command>	Enter the action to execute when the condition above is met.

Context IP Events

Triggers an action script based on IP address state change, i.e when IP address/interface is up or down.

Step	Command	Purpose
1	device(act)[<rule-name>]#[no] condition ip {address interface}:<if-name>.<label> {LINKUP LINKDOWN} [initial]	Add or remove an IP address condition to the rule, where <if-name> is the IP interface name and <label> is the IP address label.

SIP Gateway Events

Triggers an action script based on the SIP Registration status, i.e when device is Registered.

Step	Command	Purpose
1	device(act)[<rule-name>]#[no] condition sip gateway:<gw-name> {event-name} [initial]	<p>Add or remove SIP Gateway condition to the rule, where <gw-name> is the Context SIP-Gateway name.</p> <p>Possible events are:</p> <p>NOTIFY_CHECK_SYNC_NOLOAD - See Chapter 49, section "SIP Notify Check-Sync Event" for more information</p> <p>NOTIFY_CHECK_SYNC_RELOAD - See Chapter 49, section "SIP Notify Check-Sync Event" for more information</p> <p>NOT_REGISTERED - Event will come if the last registration unregistered or expired for the specified gateway</p> <p>REGISTERED - Event will come if the first registration was successful for the specified gateway</p>

System NTP Events

Triggers an action script based on NTP events, i.e when the system NTP server initially sets the time.

Step	Command	Purpose
1	device(act)[<rule-name>]#[no] condition system ntp { TIME_INITIALIZED TIME_SYNCHRONIZED } [initial]	<p>Add or remove a timer condition to the rule. The first time the NTP server sets the time, the system ntp TIME_INITIALIZED is triggered. System ntp TIME_SYNCHRONIZED is then triggered when ntp reaches, for the first time, the synchronized state.</p>

System Timer Events

Triggers an action script using the system timer.

Note A system timer must already be configured in order to execute the commands below. See **Chapter 7, section “Timed Execution of CLI Command”** for more information.

Mode: actions

Step	Command	Purpose
1	device(act)[<rule-name>]#[no] condition system timer:<timer-name> TIMEOUT [initial]	Add or remove a timer condition to the rule.

Mode: configure

Step	Command	Purpose
1	device(cfg)#[no] timer <name> ... trigger-event-timeout	Create a timer that generates an action TIMEOUT event.

Actions

Defines a CLI Command script that will be executed when the condition events occur.

Mode: actions

Step	Command	Purpose
1	device(act)[<rule-name>]# [no] action [<index>] <CLI_script>	<ul style="list-style-type: none"> • Can be multiline. ([<index>] defines line number). For more help on indexing and list operations, see ACL help. • <CLI_script> commands will be executed as if they were typed one-by-one in configuration mode.

Example:

```
[node](act)[rule1]# action "port bri 0 0"
[node](act)[rule1]# action shutdown
```

The example above will have the same effect as if user typed the command below in configuration mode.

```
port bri 0 0
shutdown
```

Note Observe the use of quotes to write multi-word commands.

Chapter 6 **System Image Handling**

Chapter contents

Introduction	85
System image handling task list	85
Displaying system image information	85
Displaying Update Status Information	86
Copying system images from a network server to flash memory	86
Switch to the inactive image	87
Erase inactive image on dual-image system	87

Introduction

System image handling management is a complex and feature rich system allowing a user to perform various upgrades on the devices. It allows a user to perform full upgrades and partial upgrades. It allows you to upgrade system configuration(s) seamlessly. The upgrades tasks are supported both from the CLI and WMI. You can copy files to flash from TFTP and local flash space. You can also upgrade from HTTP.

System image handling task list

To load and maintain system images, perform the tasks described in the following sections:

- Displaying system image information
- Displaying Update Status Information
- Copying system images from a network server to the Flash memory
- Erase inactive image on dual-image system
- Switch to the inactive image

Displaying system image information

This procedure displays information about system images and driver software.

Mode: Administrator execution

Step	Command	Purpose
1	device# show system image	Lists the system software release version, information about optional interface cards mounted in slots and other information that is the currently running system software. If you have just completed a download of new system software from the tftp server, you must execute the reload command in order to be running with the new system software. This applies equally to driver software. In some cases, the device may reboot itself.

```
device(cfg)#show system image
Software Image #1
=====
Image State : active, next
Build Version : 3.5.7-15061
Build Date : 2015/05/01
Build Number : 15061
Build Type : Release
Software Image #2
=====
Image State : inactive
Build Version : 3.4.2
Build Date : 2014/03/26
Build Number : 1
Build Type : Release
```

Step	Command	Purpose
1	device#show system version or device#show version	Show Software and Hardware versions. "show version" is a shortcut command for "show system version"

```
device(cfg)#show system version
Software Version      : 3.9.2-15121
Hardware Version      : 2
Hardware Revision     : 1
```

```
device(cfg)#show version
Software Version      : 3.9.2-15121
Hardware Version      : 2
Hardware Revision     : 1
```

Displaying Update Status Information

Mode: Operator/Admin execution

Step	Command	Purpose
1	device>show update status [continuously]	Shows the current update status.
2	device>show update progress	Shows the current update progress. This command is only available if the upgrade was manually started (e.g. by entering copy tftp://. flash:command). Enter CTRL-C to break the command.

```
device>show update status
System update status
=====
```

```
Status:                not updated
Last state:            done
```

```
device>show update progress
% No upgrade in progress
```

Copying system images from a network server to flash memory

As mentioned previously, the system image file contains the application software that runs Trinity; it is loaded into the flash memory at the Patton Electronics Co. factory. Since most of the voice and data features of the Device are defined and implemented in the application software, upgrading to a new release might be necessary if you want to have additional voice and data features available. A new system image file must be stored permanently into the flash memory of your Device to be present when booting the device. Since the system image file is preloaded at the Patton Electronics Co. factory, you will have to download new Trinity application software

only if a major software upgrade is necessary or if recommended by Patton Electronics Co. Under normal circumstances, downloading a system image file should not be needed.

Downloading a new system image file means storing it permanently at a defined location within the Patton device flash memory. To store the system image file, you must use a special download image bundle file. This bundle file contains directions for the system that describe how to handle the system image file and where to store it. The direction for the system upgrade contained in a file called manifest which is a part of the upgrade image.

Mode: Administrator execution

Step	Command	Purpose
1	device(cfg)# copy tftp://<hostname>/<file> flash: [reload no-image-switch]	Downloads the image file from the TFTP server at address <hostname> and starts the system image download process. This progress is visualized with a progress bar, printing dots according to the time elapsed since the start of each upgrade operation. “ reload ” parameter performs a reboot of the system when the upgrade is successful and the image is switched “ no-image-switch ” will only perform an upgrade without switching the image (old copy to flash command “ <i>flash-cfg.</i> ” behavior)

Switch to the inactive image

This new command switches to the inactive image (only available on dual image system).

Step	Command	Purpose
1	device(cfg)# system image switch	Switches to the inactive image upon a reload.

Erase inactive image on dual-image system

This new command will erase the inactive partition of a dual-image system. The status of the erased image is set to “invalid”.

Step	Command	Purpose
1	device# erase system image inactive	Erases the inactive image

Chapter 7 **Configuration File Handling**

Chapter contents

Introduction	89
Understanding Configuration Files	89
Shipping Configuration	90
Configuration File Handling Task List	90
Copying Configurations Within the Local Memory	91
Replacing the Startup Configuration with a Configuration from Flash Memory	92
Copying Configurations To and From a Remote Storage Location	93
Replacing the Startup Configuration with a Configuration Downloaded from TFTP Server	93
Displaying Configuration File Information	94
Modifying the Running Configuration at the CLI	95
Modifying the Running Configuration Offline	96
Deleting a Specified Configuration	97

Introduction

This chapter describes how to upload and download configuration files to and from a Trinity device. This chapter also describes some aspects of configuration file management. Refer to chapter 6, “[System Image Handling](#)” on page 84 for more information.

This chapter includes the following sections:

- Shipping configuration (see page 90)
- Configuration file handling task list (see page 90)

All Patton devices are shipped with a configuration file installed in the factory, which is stored in their flash memory.

A configuration file is like a script file containing Trinity commands that can be loaded into the system. Configuration files may also contain only partial configurations. This allows you to keep a library of command sequences that you may want to use as required. By default, the system automatically loads the shipping configuration from the flash memory if no user-specific configuration is defined as the startup configuration.

Changing the current running configuration is possible as follows:

You may change the running configuration interactively. Interactive configuring requires that you access the CLI by using the **enable** command to enter administrator execution mode. You must then switch to the configuration mode with the command **configure**. Once in configuration mode, enter the configuration commands that are necessary to configure your Patton device.

- You can also create a new configuration file or modify an existing one offline. You can copy configuration files from the flash memory to a remote server. Transferring configuration files between the flash memory and a remote system requires the Trivial File Transfer Protocol (TFTP). The TFTP server must be reachable through one of the Patton device network interfaces.

See Chapter 4, “[Accessing the CLI](#)” on page 67 for information concerning access to the CLI.

The following sections focus on Trinity memory regions and software components that can be copied within the memory or uploaded/downloaded between a TFTP server and the memory of the Patton device. Since Trinity uses a specific vocabulary in naming those software components, refer to appendix “[Trinity Architecture Terms and Definitions](#)” on page 923 to ensure that you understand the concepts. Refer to chapter 6, “[System Image Handling](#)” on page 84 for a brief description of how Trinity uses system memory.

Understanding Configuration Files

Configuration files contain commands that are used to define the functionality of Trinity. During system startup, the command parser reads the factory or startup configuration file command-by-command, organizes the arguments, and dispatches each command to the command shell for execution. If you use the CLI to enter a command during operation, you alter the running configuration accordingly. In other words, you are modifying a live, in-service system configuration.

```
bind interface LAN router
  no shutdown

port ethernet 0 0
medium auto
encapsulation ip
bind interface LAN router
no shutdown

port ethernet 0 1
medium 10 half
encapsulation ip
bind interface WAN router
no shutdown
```

Figure 7. Sample configuration file

Each configuration file stored in the flash memory needs a unique name. The user has to assign a file name to any user-specific configuration. Trinity predefines some names for configuration files. These are the shipping configuration (*shipping-config*), startup configuration (*startup-config*), minimal configuration (*minimal-config*) and running configuration (*running-config*) file names. Refer to appendix A, “Trinity Architecture Terms and Definitions” on page 923 to learn more about configuration file types.

Shipping Configuration

Patton devices are delivered with a *shipping configuration* in the logical region *config*. This shipping configuration initially parameterizes the most useful network and component settings of Trinity.

Once a user-specific configuration is created and stored as the startup configuration, the shipping configuration is no longer used, but still remains in the persistent memory. It is possible to switch back to the shipping configuration at any time during the operation of a Patton device configuration. The getting started guide included with your Patton device describes the restoration procedure for restoring the default settings.

Configuration File Handling Task List

This section describes how to create, load, and maintain configuration files. Configuration files contain a set of user-configured commands that customize the functionality of your Patton device to suit your own operating requirements.

The tasks in this chapter assume that you have at least a minimal configuration running on your system. You can create a basic configuration file by using the **configure** command; see section “Modifying the Running Configuration at the CLI” on page 95 for details.

To display, copy, delete, and download or upload configuration files, perform the tasks described in the following sections:

- Copying configurations within the local memory (see [page 91](#))
- Replacing the startup configuration with a configuration from the Flash memory (see [page 92](#))
- Copying configurations to and from a remote storing location (see [page 93](#))
- Replacing the startup configuration with a configuration downloaded from the TFTP server (see [page 93](#))

- Displaying configuration file information (see [page 94](#))
- Modifying the running configuration at the CLI (see [page 95](#))
- Modifying the running configuration offline (see [page 96](#))
- Deleting a specified configuration (see [page 97](#))

Copying Configurations Within the Local Memory

Configuration files may be copied into the local memory in order to switch between different configurations. Remember the different local memory regions in Trinity as shown in [figure 8](#).

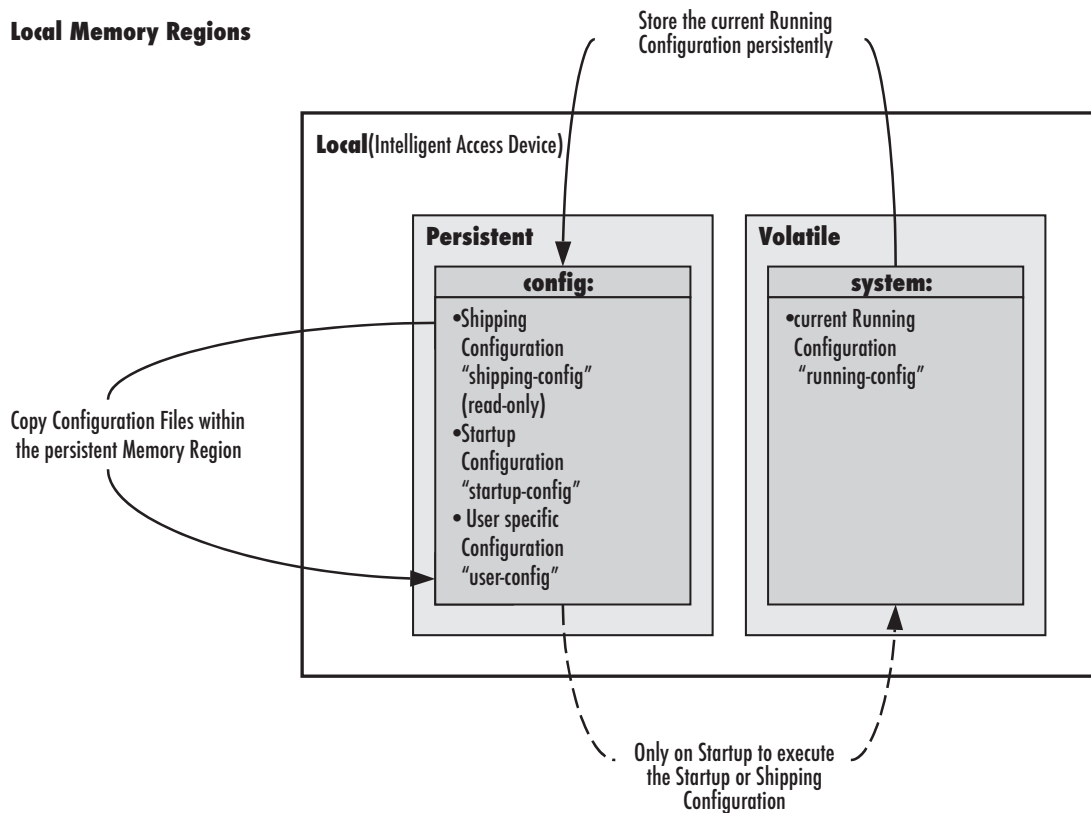


Figure 8. Local memory regions

In most cases, the interactively modified running configuration known as the *running-config*, which is located in the volatile memory region *system:*, is copied into the persistent memory region *config:*. This running config is stored under the name *startup-config* and replaces the existing startup configuration.

You can copy the current running configuration into the persistent memory region *config:* under a user-specified name, if you want to preserve that configuration.

In addition, an already existing configuration is usually copied into the persistent memory region *config:* by using a user-specified name, for conservation or later activation.

As shown in [figure 8](#) the local memory regions are identified by their unique names, like *config:*, which is located in flash memory, and *system:*, which is the system RAM, i.e. the volatile memory. As already mentioned,

configuration files in the same memory region need a unique name. For example, it is not possible to have two configuration files with the name *running-config* in the memory region *config*.

As you might expect, the **copy** command does not move but replicates a selected source to a target configuration file in the specified memory region. Therefore the source configuration file is not lost after the copy process. There are four predefined configuration file names for which it is optional to specify the memory region, namely *shipping-config*, *startup-config*, *minimal-config* and *running-config*.

Mode: Administrator execution

Step	Command	Purpose
1	device#copy {shipping-config startup-config minimal-config running-config config: source-name } config: target-name	Copies the selected source configuration file <i>source-name</i> as target configuration file <i>target-name</i> into the local memory.

Example: Backing up the startup configuration

The following example shows how to make a backup copy of the startup configuration. It is copied under the name *backup* into the flash memory region *config*.

```
device#copy startup-config config:backup
```

Replacing the Startup Configuration with a Configuration from Flash Memory

It is possible to replace the startup configuration by a configuration that is already present in the flash memory. You can do so by copying it to the area of the flash memory where the startup configuration is stored.

Mode: Administrator execution

Step	Command	Purpose
1	device# copy config:backup startup-config	Replaces the existing persistent startup configuration with the startup configuration <i>backup</i> already present in flash memory.

Note The configuration *backup* can be a previously backed up configuration or previously downloaded from a TFTP server.

Copying Configurations To and From a Remote Storage Location

Configuration files can be copied from local memory (persistent or volatile region) to a remote data store. From within Trinity, the remote TFTP server is represented by the memory region *tftp:* in combination with the IP address of the TFTP server and the name and path of the configuration file. We will explain the usage of the remote memory region *tftp:* in the following section more detailed. Another typical task is uploading the current running configuration to the remote data store for backup purpose, or if an extensive configuration file is to be edited on the remote host. In this case the running configuration, named *running-config*, which is to be found in the volatile memory region *system:* is transferred to the TFTP server. On the TFTP server the running configuration is stored to a file whose name is defined as one of the arguments of the **copy** command.

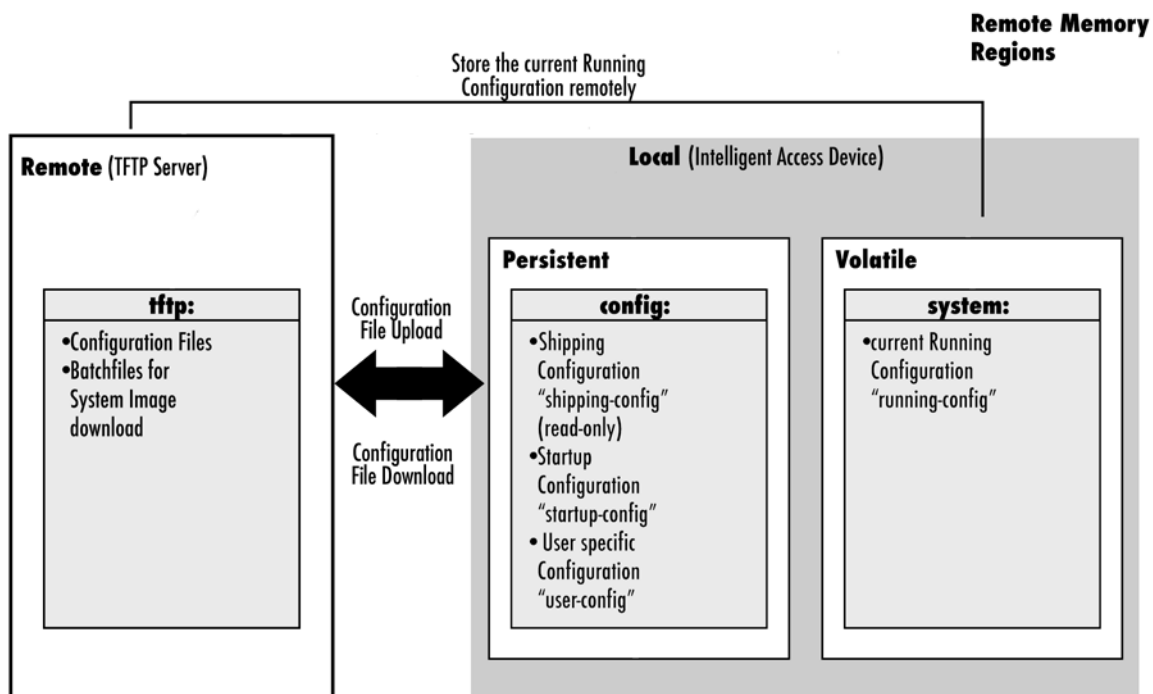


Figure 9. Remote memory regions for Trinity

Finally, configuration files, i.e. the startup configuration or a user-specific configuration that is stored in the persistent memory region *config:* are often uploaded to the remote data store for backup, edit or cloning purposes. The latter procedure is very helpful when you have several Patton devices, each using a configuration which does not greatly differ from the others, or which is the same for all devices. During the configuration of the first Patton device according to your requirements, the running configuration of this device, named *running-config* and located in the volatile memory region *system:*, is edited. Next, the configuration is tested and if everything is as required, the running configuration is copied as startup configuration, named *startup-config*, into the persistent memory region *config:* of the target device. After this, the startup configuration is transferred to the TFTP server, where it can be distributed to other Patton devices. These devices therefore get clones of the starting system if the configuration does not need any modifications.

Replacing the Startup Configuration with a Configuration Downloaded from TFTP Server

From within the administration execution mode, you can replace the startup-configuration by downloading a configuration from the TFTP server into the flash memory area where to store the startup configuration.

Mode: Administrator execution

Step	Command	Purpose
1	device(cfg)# copy tftp://server-ip-address/[:port]/new-startup config:startup-config	Downloads the configuration file <i>new-startup</i> from the TFTP server at address <i>ip-address</i> replacing the existing persistent startup configuration. Optionally you can enter the UDP <i>port</i> where the TFTP server listens. If the port is not specified, the default port 69 is used. This progress is visualized with a counter, counting up from 0 to 100% according to the downloaded amount of the file size. Should the download fail, an error message <i>% File Transfer - Get failed</i> is displayed.

Example: Sample configuration download from the TFTP server

The following example shows how to replace the persistent startup configuration in the flash memory of a Patton device by overwriting it with the configuration contained in the file *new-startup* located on the TFTP server at IP address 172.16.36.80.

1. Download the startup configuration with the **copy** command into the flash memory area where to store the startup configuration.

```
device>enable
device#configure
device(cfg)#copy tftp://172.16.36.80/user/new-startup config:startup-config
Download...100%
device(cfg)#
```

2. Check the content of the persistent startup configuration by listing its command settings with the **show** command.

```
device#show config:startup-config
```

Displaying Configuration File Information

This procedure describes how to display information about configuration files

Mode: Administrator execution

Command	Purpose
show config:	Lists all persistent configurations
show running-config	Displays the contents of the running configuration file
show startup-config	Displays the contents of the startup configuration file
show running-config current-mode	Displays only the running-config of the current mode.
show running-config "<some mode>"	Displays the running-config of any named mode



It is recommended that you *never* save a configuration in startup-config or a user-specific configuration with the **cli config defaults** command because the additional list of default commands consumes significant portions of the *config:* memory.

Note Application files can be very long when displayed (by using the **show** command). To make them easier to read, many default commands are not displayed when executing the **show running-config** command. However, the administrator may want to see the entire configuration, including these normally “hidden” default commands. To see all commands, execute the **cli config defaults** command. By issuing a **show running-config** command afterwards, you will see all the commands, a list which is significantly longer. To hide these hidden commands again, issue the **no cli config defaults** command.

Modifying the Running Configuration at the CLI

Trinity accepts interactive modifications on the currently running configuration via the CLI. Interactive configuring needs access to the CLI. Use the **enable** command to enter administrator execution mode, and then switch to the configuration mode by typing the command **configure**. Once in configuration mode, you can enter the configuration commands that are necessary to your Patton device’s operation. When you configure Trinity by using the CLI, the shell executes the commands as you enter them.

When you log in using the CLI, all commands you enter directly modify the running configuration located in the volatile memory region *system:* (or RAM) of your device. Because it is located in volatile memory, to be made permanent, your modifications must be copied to the persistent (non-volatile) memory. In most cases you will store it as the upcoming startup configuration in the persistent memory region *config:* under the name *startup-config*. On the next start-up the system will initialize itself using the modified configuration. After the startup configuration has been saved to persistent memory, you have to restart the device by using the **reload** command to cause the system to initialize with the new configuration.

The execution command **reload** accepts with the following option:

- forced—reloads the system without prompting for confirmation or for saving the running-configuration (no need to type *yes* or *no*). The question whether to save the running-configuration is automatically answered with *no*, the question whether to reload or not with *yes*.
- graceful—reloads the system only if no voice calls are ongoing. If there are voice calls, the system waits until they all are closed to reload.

Mode: Administrator execution

Step	Command	Purpose
1	device#configure	Enters administrator configuration mode
2	Enter all necessary configuration commands.	
3	device(cfg)#copy running-config startup-config	Saves the running configuration file as the upcoming startup configuration
4	device(cfg)#reload [graceful forced]	Restarts the system

Example: Modifying the running configuration at the CLI

The following example shows how to modify the currently running configuration via the CLI and save it as the startup configuration.

```
device#configure
device(cfg)#...
device(cfg)#copy running-config startup-config
device(cfg)#reload
Press 'yes' to restart, 'no' to cancel: yes
The system is going down
```

Modifying the Running Configuration Offline

In cases of complex configuration changes, which are easier to do offline, you may store a configuration on a TFTP server, where you can edit and save it. Since the Patton device is acting as a TFTP client, it initiates all file transfer operations.

First, upload the running configuration, named *running-config*, from the Patton device to the TFTP server. You can then edit the configuration file located on the TFTP server by using any regular text editor. Once the configuration has been edited, download it back into the device as upcoming startup configuration and store it in the persistent memory region *config*: under the name *startup-config*. Finally, restart the Patton device by using the **reload** command to activate the changes.

Mode: Administrator execution

Step	Command	Purpose
1	device#copy running-config tftp://device-ip-address[:port]/current-config	Uploads the current running configuration as file <i>current-config</i> to the TFTP server at address <i>device-ip-address</i> . Optionally you can enter the UDP <i>port</i> where the TFTP server listens. If the port is not specified, the default port 69 is used. This progress is visualized with a counter, counting up from 0 to 100% according to the downloaded amount of the file size. If the upload should fail an error message “% File Transfer - Put failed” is displayed.
2		Offline editing of the configuration file <i>current-config</i> on the TFTP server using any regular text editor.
3	device#copy tftp://device-ip-address/current-config config: startup-config	Downloads the modified configuration file <i>current-config</i> from the TFTP server at address <i>device-ip-address</i> into the persistent memory region <i>config</i> : by using the name <i>startup-config</i> . This progress is visualized with a counter, counting up from 0 to 100% according to the downloaded amount of the file size. Should the download fail, an error message “% File Transfer - Get failed” is displayed.
4	device#reload	Restarts the system

Example: Modifying the running configuration offline

The following example shows how to upload the running configuration from the Patton device to the file *current-config* on a TFTP server at IP address 172.16.36.80. The uploaded configuration file is written into the root directory specified by the TFTP server settings, and overwrites any existing file with the same name. Read

your TFTP server manual to get a thorough understanding of its behavior. After this, the configuration file is available for offline editing on the TFTP server. Once the configuration file *current-config* has been modified, it is downloaded from the TFTP server, at IP address 172.16.36.80, into the persistent memory region *config:* using the name *startup-config*. It will become active after a reload.

```
device#copy running-config tftp://172.16.36.80/user/current-config
Upload...100%
```

At this point in time, the offline editing of the configuration file *current-config* on the TFTP server takes place.

```
device#copy tftp://172.16.36.80/user/ current-config config:startup-config
Download...100%
device#reload
Press 'yes' to restart, 'no' to cancel: yes
The system is going down
```

Deleting a Specified Configuration

This procedure describes how to delete configuration files from the Patton device flash memory region *config:*.

Mode: Administrator execution

Step	Command	Purpose
1	device#show config:	Lists the loaded configurations
2	device#erase config:name	Deletes the configuration <i>name</i> from the flash memory.

Example: Deleting a specified configuration

The following example shows how to delete a specific configuration from among a set of three available configurations in Flash memory. The configuration named *minimal* is to be deleted, since it is no longer used.

1. Use the command **show config:** to list all available configurations.

```
device#show config:
Persistent configurations:
backup
minimal
startup-config
shipping-config
```

2. Delete the configuration named *minimal* explicitly.

```
device#erase config:minimal
```

3. Enter again the command **show config:** to check if the selected configuration was deleted successfully from the set of available configurations.

```
device#show config:
Persistent configurations:
backup
startup-config
shipping-config
```

Chapter 8 System Licensing

Chapter contents

Introduction	99
Install licenses permanently	101
Installing licenses	101
Inspecting installed licenses	102
Erasing installed licenses	105
Configure a Patton device as License Server	105
Prepare the system	107
Configure the NodeMS-Server	108
Create organizations for tenants	108
Create sub-organizations within an organization	110
Define which and how many licenses a particular device is able to lease	112
Create a nodems-device profile	113
Link devices to a nodems-device profile	115
Inspect the license-lease database	116
Configure Patton devices to lease licenses from a License Server	121
Prepare the system	121
Connect a Patton device to a License Server	122
Display information about the License Server connection	124

Introduction

Some Trinity features are locked and can be used only if the device has access to a corresponding feature license. As depicted in [figure 11](#) there are three different ways feature licenses are made available to a Patton device:

- Built-in licenses
- Installed licenses (installed by the factory or by the user)
- Licenses leased from a License Server

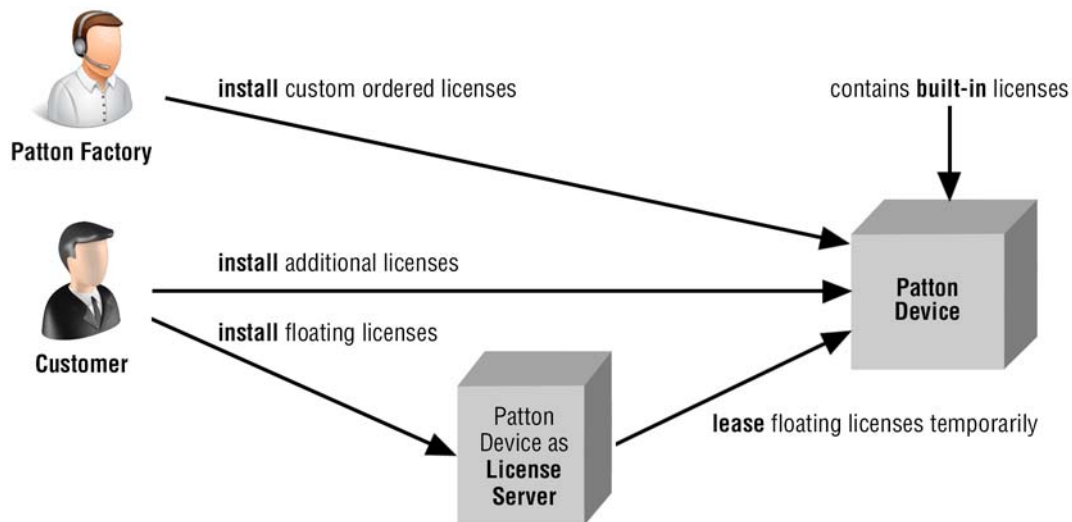


Figure 10. How licenses may be made available to Patton devices.

Built-In licenses are part of the device hardware and cannot be erased or moved to another device. Each model comes with a different set of built-in licenses.

Installed licenses enter the device in two scenarios:

- If you order licenses together with a device the Patton factory pre-installs the ordered licenses prior to shipping it to you.
- If you need additional licenses, you may order and install them yourself. Installed licenses are bound to a specific device and cannot be moved to another device.

Alternatively, you may install licenses on a License Server that **leases** floating licenses temporarily to your devices. This approach is required for Trinity running in a virtual machine but available also for physical devices (e.g. eSBCs, VoIP gateways, IP routers, etc.).

To install licenses on your devices or on the License Server, there are two methods for unlocking features as depicted in [figure 11](#) and [figure 12](#) on page 100.

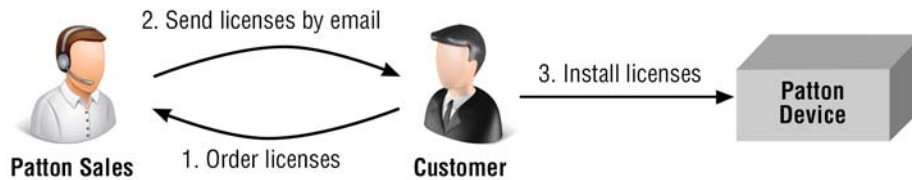


Figure 11. Install licenses permanently on the device

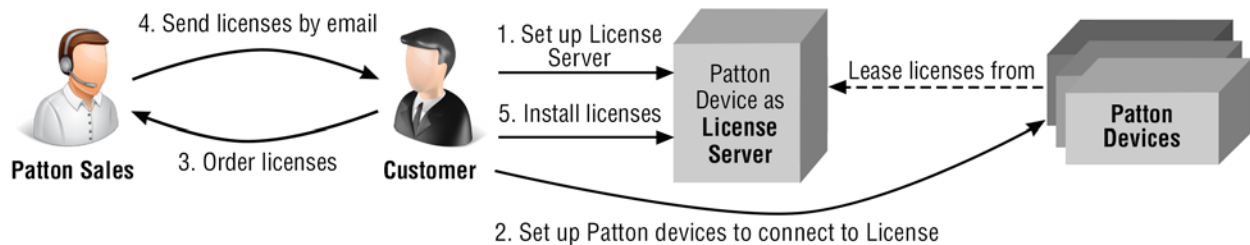


Figure 12. Device leases floating licenses from a license server

Note Features on a virtual Trinity device (Trinity running in a virtual machine) can only be unlocked with a License Server. It is not possible to install licenses permanently on VMs. Running a Licenses Server for non-virtual Patton devices (e.g. eSBCs, VoIP Gateways, IP Routers, etc.) is possible as well. This brings the following advantages:

- You don't have to order licenses for each device separately.
- Licensed quantities such as the number of SIP sessions can be shared among multiple devices and distributed based on the current needs of the devices.

Note Licenses installed on the device or leased from a License Server are added on top of built-in licenses that are shipped with the device.

This chapter includes the following sections:

- Install licenses permanently (on Patton devices or on the License Server) (see [page 101](#))
- Configure a device as License Server (see [page 105](#))
- Configure Patton devices to lease licenses from a License Server (see [page 121](#))

If you want to install licenses permanently on your devices, please read the first section. If you plan to run a License Server to distribute floating licenses you can first read the latter two sections. The first section however is important as well to install the floating licenses on the License Server.

Install licenses permanently

Installing a license permanently on a Patton device unlocks the corresponding licensed feature on that device. The advantage of this static and permanent assignment of licenses to a device is that no License Server is required. The disadvantage in a larger network of Patton devices is that you have to order and install licenses for each device separately.

If you run a License Server to distribute licenses dynamically, this step is still needed on the License Server to permanently install floating licenses that will be distributed dynamically to the other Patton devices over the IP network.

Licenses are installed, erased, and inspected by CLI commands as described below.

Installing licenses

If you order licenses from Patton, you will receive an e-mail with a set of CLI commands - one command for each feature. Copy the commands and paste them into an open Telnet, SSH, or console connection. This unlocks the corresponding licensed feature. The following procedure applies for both installing permanent licenses to Patton devices and installing floating licenses to a License Server:

Procedure: Installing licenses

Mode: Operator Exec

Command	Purpose
<code>node > install license encrypted-license</code>	Installs the specified license permanently.

Example: Install a license as received by e-mail from Patton

```
node>install license 10011002R1Ws63yKV5v28eVmhdsvGj/JwKqIdpC4Wr1BHaN- tenXUYF/
2gNLoihifacaTPLKcV+uQDG8LJis6EdW6uNk/ GxVObDEwPFJ5bTV3bIIfUZ1eUe+8c5OpC-
Cd7PSAe83Ty2c/ CnZPS1EjIrVlJrr8VhOr1DYxkEV9evBp+tSY+y9sCeXhDWt5Xq15SAP1znTLQmym7f-
Dakvm+zltzswX/
KXl3sdkR0ub9IX4Sjn6YrvkyrJ2dCGivTTB3iOBmRjV1u
```

Note Licenses are bound to a specific device. When ordering a license, you have to specify the serial number (and the model) of the device. The license sent to you by Patton can only be installed on that particular device and cannot be migrated to another device.

Note If Trinity is running in a Virtual Machine (VM), you cannot install licenses permanently, because there is no unique hardware identifier and VMs can be cloned. Instead, you have to run a separate License Server and configure the virtual Trinity instance to lease licenses from that server. Also note that each virtual Trinity instance requires a "virtual-instance" license to run. That is, you need as many "virtual-instance" licenses as the total number of VM instances you plan to launch.

Note For most license types, no reboot is required. However, for the following licenses you have to reboot the device to unlock the feature: *fips*, *openvpn*, *license-server*.

Inspecting installed licenses

After installing licenses, you can check if they are successfully added to your system using the following command:

Procedure: Show installed licenses

Mode: Operator Exec

Command	Purpose
<code>node > show system licenses [license-name]</code>	<p>If the license-name argument is omitted, this command shows all available licenses in a tabular structure. Each license is printed on a separated row. The columns have the following meaning:</p> <p>License Name: Name of the licensed feature</p> <p>Source: Shows where the license is coming from.</p> <ul style="list-style-type: none"> • Blt-In: Number of build-in licenses, pre-installed in the factory as part of every ordered device. • Inst.: Number of licenses installed permanently by the user (see license command above). Also includes licenses installed in the factory as part of an order option. • Leased: Number of licenses leased temporarily from a License Server. <p>Available: Shows how many licenses are available to the device in total.</p> <ul style="list-style-type: none"> • Total: Current number of licenses available to the device. <p>Usage: How many of the total licenses are currently used by system components and how many are still available.</p> <ul style="list-style-type: none"> • Alloc: Number of licenses in use by the device's components. • Free: Number of licenses still available. <p>If the license-name argument is given, the command prints details about the specified license.</p>

Example: Display all installed licenses on a Patton device

```
node>show system licenses
```

```
Local Licenses
=====
```

```
The following licenses are available on the local device:
Serial Number: 00A0BA09F412
```

```
-----
```

License Name	Source:			Available: Total	Usage:	
	Blt-In +	Inst. +	Leased =		= Alloc +	Free
----- sip-sessions	4		188	192	2	190
----- iprouter	Y			Y		
packetsmart		Y		Y		
sip-registrar			Y	Y		
sip-tls-srtp			Y	Y		
transcoding			Y	Y		
vpn			Y	Y		

Let us examine the *sip-sessions* license in the output above in more detail: The device already ships with 4 licenses. The device is connected to a licenses server from which it leased additional 188 licenses. Thus, the device is now able to make up to 192 calls. Currently, two sessions are active while the other 190 sessions are still available. License that don't have an assigned quantity are just printed with the letter 'Y' if they are present.

On a License Server (see section “[Configure a Patton device as License Server](#)” on page 105) the "show system licenses" command will print two sections: *Local Licenses* and *Server Licenses*:

- The *Local Licenses* can be used on the License Server itself but are never leased to connected Patton devices. The device acting as a License Server needs a permanently installed *license-server* license (see section “[Installing licenses](#)” on page 101).
- The Server Licenses are floating licenses which cannot be used on the License Server itself but which can be leased to connected Patton devices.

Example: Display all installed licenses on a License Server

```
node>show system licenses
Local Licenses
=====
```

```
The following licenses are available on the local device:
Serial Number: 00A0BA09F435
```

License Name	Source:			Available: Total	Usage:	
	Blt-In +	Inst. +	Leased =		= Alloc +	Free
----- license-server		Y		Y		

```
Server Licenses
=====
```

```
The following floating licenses are distributed to connected client
devices; they cannot be used on the local device directly:
```

License Name	Source:			Available: Total	Usage:	
	Blt-In +	Inst. +	Leased =		= Alloc +	Free

iprouter	100	100	100	
packetsmart	100	100	100	
sip-registrar	100	100	2	98
sip-sessions	10000	10000	376	9624
sip-tls-srtp	100	100	2	98
transcoding	100	100	0	100
virtual-instance	100	100	1	99
voice-channels	10000	10000		10000
vpn	100	100	2	98

The License Server in the example above has 9 different types of floating licenses installed; for example, 10000 sip-sessions licenses, 376 of which are currently leased (*Alloc* column) to connected Patton devices, the remaining 9624 which are available (*Free* column) for other devices. The License Server balances license automatically to those devices that need them most, i.e. to those devices with the least number of free licenses.

Example: Display details about the *iprouter* license

```
node>show system licenses iprouter

iprouter
=====

ID:                               1
Description:                       IP Routing
Device Supports Resource:         yes
Device Needs Resource:           yes (1 interests)

For Local Use
-----

Built-In
```

The output shows a description of the license, whether the device supports and requires a license to operate and how the devices has access to the license: Built-In means that the license has been pre-installed in the factory.

Erasing installed licenses

Installed licenses can be removed permanently. This makes the licensed feature unavailable for the device. Note that built-in licenses or licenses leased from a License Server cannot be erased.

Procedure: Erasing installed licenses

Mode: Operator Exec

Command	Purpose
<code>node > erase license license-name [local floating all]</code>	Erases an installed license permanently. Use the "show system licenses" command to get a list of installed licenses (see above). <ul style="list-style-type: none"> • local: Erases license the device uses for unlocking local features. This is the default behavior when omitting the licence-type. • floating: Erases floating licenses that are leased to connected client devices • all: Erases both local and floating licenses

Example: Permanently erases the *iprouter* license from the Patton device. The device will stop working as IP router.

```
node>erase license iprouter
```

Note For most license types, no reboot is required. However, for the following license you have to reboot the device to lock the feature: *fps*, *openvpn*, *license-server*.

Configure a Patton device as License Server

A License Server is a Patton device running Trinity, dedicated to the distribution of floating licenses to other Patton devices. On the License Server, you still have to install floating licenses with the procedure described in the previous section. But those floating licenses (e.g. 10000 *sip-session* licenses) are then available to all Patton devices that are connected to the License Server. The License Server supports the following features:

- Data security between the client devices and the License Server is provided by a secure connection (TLS).
- A key that is shared between the License Server and the connecting devices makes sure that licenses are only leased to "trusted devices".
- You are able to configure how licenses are distributed to client devices by partitioning your network into different *organizations* and *sub-organizations*. For example, if you own 10000 *sip-session* licenses, you could create two sub-organizations *Productive* and *Testlab* and dedicate 9000 licenses to devices in the *Productive* sub-organization and 1000 licenses to devices in the *Testlab*.
- Furthermore, you can define different device profiles and specify how many licenses each device model is able to lease (e.g. 15 *sip-sessions* for devices with one PRI port).

- Instead of assigning each device model a fixed number of licenses you can configure the License Server to assign them dynamically based on the current need of the devices. In this case licenses (e.g. sip-sessions) are automatically balanced to those devices with the least number of free licenses.
- You are free to configure this assignment of licenses to devices either on the License Server or on the individual client devices.

Example: Set up a simple license server with one organization (MY_ORG) and one sub-organization (MY_SUBORG) that distributes the installed floating licenses dynamically to all connected devices:

```
profile nodems-device DEFAULT
  resource any

nodems-server

  organization MY_ORG

    sub-organization MY_SUBORG
      organization-key 0ABF8445ADF32D563FFE8D3564DCAB59

      device any
        use profile nodems-device DEFAULT

nodems-server
  no shutdown
```

The *DEFAULT* *nodems-device* profile defines a default device model that allows all matching devices to lease any type of resources (=licenses) without limitation.

The *nodems-server* mode contains the configuration commands for the License Server¹. The *nodems-server* mode contains an organization, which contains a sub-organization. The *any* device mode within the *MY_SUBORG* sub-organization contains the configuration for all devices not listed explicitly. You may list devices by their serial number to assign them other resource profiles.

The corresponding configuration on the client devices that connect to the License Server is as follows:

```
nodems-client
  server http://license-server.example.com
  organization-key 0ABF8445ADF32D563FFE8D3564DCAB59
  resource any
  no shutdown
```

The organization-key configured on the client device is used for both authenticating the device to the license server and linking the device to a configured organization/sub-organization configuration on the server. Organizations and sub-organizations can be used to assign different license-distributing policies to connecting devices. [Figure 13](#) on page 107 shows a License Server that distributes its installed 10000 floating *sip-sessions* licenses to two companies, reflected in the configuration as *organizations*. The devices of the first company are split up into two *sub-organizations*, a test lab and a productive network. Each organization and sub-organization may be configured with license restrictions. In this example, all devices in the test lab together may lease up to 1000 *sip-sessions* licenses, whereas all devices in the productive sub-organization may lease up to 5000

1. NodeMS = Node-Management System. In the future, we will add more remote-management features to this mode.

licenses. The *DEFAULT* sub-organization of the organization *COMPANY2* has no restriction configured. Based on the restrictions of the other organization all devices of the second company together may lease up to $10000 - 1000 - 5000 = 4000$ *sip-sessions* licenses.

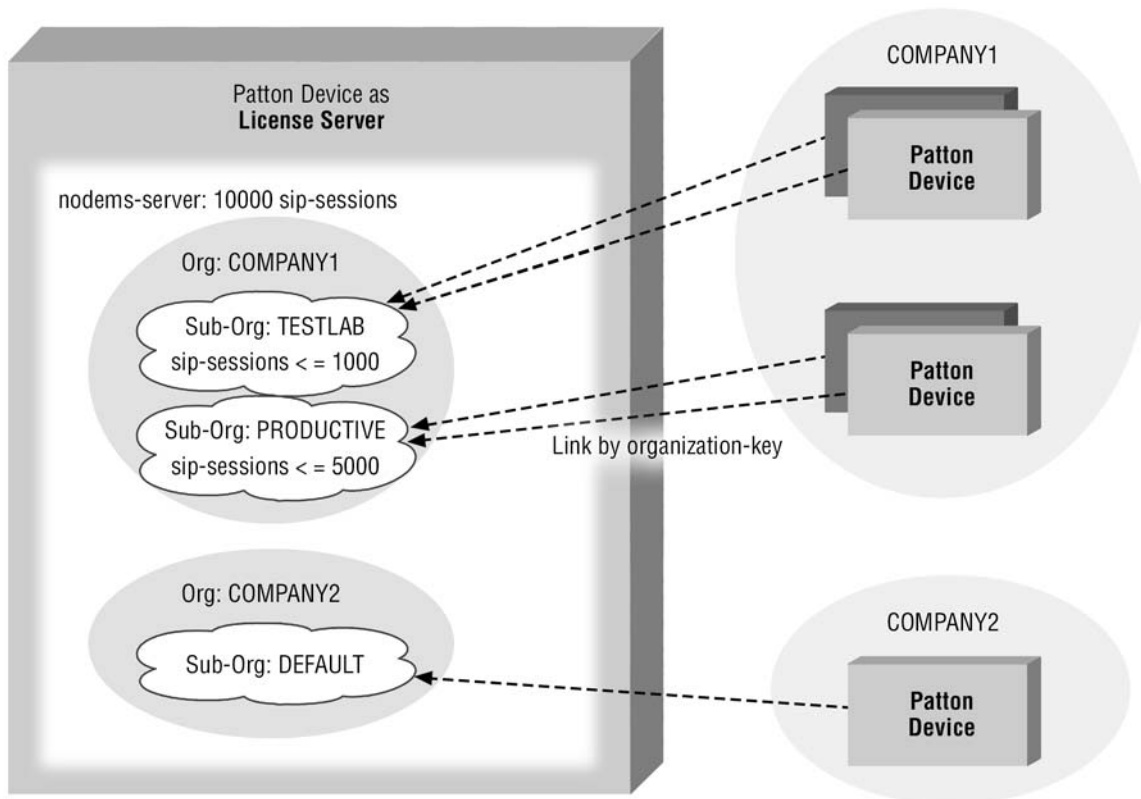


Figure 13. Connected Devices are grouped to Organizations and Sub-Organizations

Prepare the system

Before you configure the License Server you have to prepare the system as follows:

1. Make sure the server is reachable from the client devices, e.g. configure network interfaces and routing. If you have configured an ACL (Access-Control List), enable incoming connections to TCP port 443 (the default HTTPS server port) (see chapter 38, "Access Control List Configuration" on page 440).
2. Make sure the server's clock is accurate. The best way to grant this is to get the clock updates from an NTP server (see chapter . This is required because the License Server generates a server certificate to authenticate itself to the client devices; this certificate is valid from the time you start the server. If the clock is inaccurate, client devices may refuse connecting to the server.
3. After configuring the License Server, install the floating licenses received from Patton according to the procedure described in section "Install licenses permanently" on page 101.

Configure the NodeMS-Server

Procedure: Configure the NodeMS-Server

Mode: Configure

Command	Purpose
<i>node(cfg)# nodems-server</i>	Enters the NodeMS-Server configuration mode. The license-distribution system is part of Patton's Node-Management System (NodeMS). In the future, more services will be added to manage Patton devices remotely over one secure communication channel.
<i>node(nodems-srv) # [reset] port port</i>	Configures the TLS/TCP <i>port</i> number on which the NodeMS server listens for incoming connections from client devices. The default port number is 443. The server listens on all active network interfaces.
<i>node(nodems-srv) # [no] shutdown</i>	Starts or stops listening for incoming connections from client devices. "shutdown" drops all client connections.

Create organizations for tenants

In a multi-tenant setup, an organization is typically used to group devices of one tenant (company) and apply license-restriction policies to the entire group.

Procedure: Create an organization

Mode: NodeMS-Server

Command	Purpose
<i>node(nodems-srv) # [no] organization orga- nization-name</i>	Creates/Destroys an organization and enters its configuration mode. You have to generate at least one organization for the system to work.

Command	Purpose
node(org)[organization-name] # [reset] allocate (auto lease for amount (minutes hours days weeks) permanent)	<p>Configures whether the licenses are installed persistently on the connecting client devices (not supported yet) or leased temporarily to the client device in this organization and for how long. Typically, this configuration should not be changed: the default lease time is one day.</p> <ul style="list-style-type: none"> • lease for: Licenses are leased for the specified period of time. If the client device loses contact to the License Server it is able to continue using the leased license for this period of time. • permanent: Licenses are permanently installed to the connected client device (not supported yet). • auto: Licenses are leased to client devices for one day. This is the default configuration. <p>A sub-organization within this organization or a device within the sub-organization may overwrite this setting for the particular sub-organization or device, respectively.</p>

Command	Purpose
node(org)[organization-name] [no] resource resource-name [amount] [floating]	<p>Configures license restrictions for this organization where each line corresponds to one license type (called "resource" here). The no-form removes the restriction. In a new organization, no license is restricted by default. That is, all floating licenses installed on the License Server are ready to be leased to connecting clients belonging to this organization.</p> <ul style="list-style-type: none"> • <i>resource-name</i>: Name of the license (e.g. <i>sip-sessions</i>). • <i>amount</i>: Number of licenses all clients in this organization together are able to lease in total. You can either specify a number (e.g. "10") or a number range (e.g. "..10" for between zero to ten, or "10..100" for between ten and hundred). When specifying a lower limit, the organization pre-allocates the specified licenses and reserves them for this organization. When specifying an upper limit, this is the maximum number of resources this organization is allowed to lease to all clients. A typical use case is to only limit the max. total number of licenses in this organization, e.g. "..1000". • floating: The amount parameter refers to the number of floating licenses leased to all devices and not to the total number of licenses on the devices. Each device may have pre-installed licenses which are not accounted for with this configuration. <p>Note License restrictions entered for the organization can be narrowed-down further in the sub-organizations. For example, if an organization has an upper limit of 1000 <i>sip-sessions</i> licenses, one sub-organization within this organization may restrict it to 100 licenses, leaving the other 900 license to the other sub-organizations.</p>

Create sub-organizations within an organization

A *sub-organization* resides within an *organization*; it groups devices of the same tenant together. Sub-Organizations are typically used if the licenses of an organization should not be available to all client devices of the tenant. For example, if the tenant runs a productive network and a test lab, he probably only wants a fraction of his licenses to be allocated to the test lab and the majority of the licenses to be reserved for the productive network. Such a configuration makes sure that the productive network always has enough licenses even if new devices are added to the test lab that potentially require more floating licenses.

Procedure: Create a sub-organization within an organization

Mode: NodeMS-Server/Organization

Command	Purpose
node(org)[organization-name] # [no] sub-organization sub-organization-name	Creates/Destroys a sub-organization and enters its configuration mode. You have to generate at least one sub-organization within each organization for the system to work.
node(sub-org)[sub-organization-name] # [no] organization-key key	Configures a secret that must be shared with all client devices that want to lease licenses from this sub-organization. By default, the system generates an organization key for each created sub-organization. You may overwrite this key but make sure that the key is unique across all organizations and sub-organization. It is possible to delete the organization-key on one sub-organization but we strongly discourage you to do so for security reasons. In that case, every device that is able to reach your License Server is able to lease licenses from it. <ul style="list-style-type: none"> • key: The same key must be configured in the nodems-client mode on the client devices to lease licenses from this sub-organization.
node(sub-org)[sub-organization-name] # [reset] allocate (auto lease for amount (minutes hours days weeks) permanent)	Configures whether the licenses are installed persistently on the client device (not supported yet) or leased temporarily to the client device and for how long. This configuration overwrites the same configuration on the parent organization and typically should not be changed; the default lease time is one day. <ul style="list-style-type: none"> • lease for: Licenses are leased for the specified period of time. If the client device loses contact to the License Server it is able to continue using the leased license for this period of time. • permanent: Licenses are permanently installed to the connected client device (not supported yet). • auto: Inherits the setting from the parent organization. This is the default configuration

Command	Purpose
node(sub-org)[sub-organization-name] # [no] resource resource-name [amount] [floating]	<p>Configures license restrictions for this sub-organization where each line corresponds to one license type (called "resource" here). The no-form removes the restriction. In a new sub-organization, no license is restricted by default. That is, all floating licenses available for the parent organization are ready to be leased to connecting clients belonging to this sub-organization.</p> <ul style="list-style-type: none"> • resource-name: Name of the license (e.g. <i>sip-sessions</i>). • amount: Number of licenses all clients in this sub-organization together are able to lease in total. You can either specify a number (e.g. "10") or a number range (e.g. "..10" for between zero to ten, or "10..100" for between ten and hundred). When specifying a lower limit, the sub-organization pre-allocates the specified licenses and reserves them for this sub-organization. When specifying an upper limit, this is the maximum number of resources this sub-organization is allowed to lease to its clients. A typical use case is to only limit the max. total number of licenses in this sub-organization, e.g. "..1000". • floating: The amount parameter refers to the number of floating licenses leased to all devices and not to the total number of licenses on the devices. Each device may have pre-installed licenses which are not accounted for with this configuration. <p>Note License restrictions entered for the parent organization can be narrowed down further in the sub-organization. For example, if an organization has an upper limit of 1000 sip-sessions licenses, one sub-organization within this organization may restrict to it to 100 licenses, leaving the other 900 licenses to the other sub-organizations.</p>

Define which and how many licenses a particular device is able to lease

NodeMS-Device profiles are used to define which device model is able to allocate which and how many licenses. Follow this procedure to assign a connecting device some licenses statically or automatically:

1. Create a nodems-device profile for the corresponding device model (e.g. eSBR with one PRI port).
2. List the device (its serial number) within a sub-organization (within an organization).
3. Link the created resource-allocated profile to this device.

Create a nodems-device profile

If you want to restrict the number of licenses that can be distributed to one particular device, we suggest that you create one nodems-device profile for each device model (that uses the same number of licenses).

Procedure: Create and configure a nodems-device profile

Mode: Configure

Command	Purpose
<i>node(cfg)# [no] profile nodems-device profile-name</i>	Creates or deletes a nodems-device profile and enters its configuration mode
<i>node(pf-ralloc)[profile-name]# [reset] allocate (auto lease for amount (minutes hours days weeks) permanent)</i>	Configures whether the licenses are installed persistently on the client device (not supported yet) or leased temporarily to the client device and for how long. <ul style="list-style-type: none"> • lease for: Licenses are leased for the specified amount of time. If the client device loses contact to the License Server it is able to continue using the leased license for this amount of time. • permanent: Licenses are permanently installed to the connected client device (not supported yet). • auto: Licenses are leased to client devices for one day. This is the default configuration.

Command	Purpose
<code>node(pf-ralloc)[profile-name]# [no] resource any</code>	<p>Specifies how many licenses are leased to one device. If the no-form of the command is used, the specified resource is not leased to connected devices at all.</p> <ul style="list-style-type: none"> • <i>resource-name</i>: Name of the license (e.g. <i>sip-sessions</i>). any is a placeholder for all licenses not specified explicitly in this profile. • <i>amount</i>: Number of licenses a client is able to lease. You can either specify a number (e.g. "10") or a number range (e.g. "..10" for between zero to ten, e.g. "10..100"). When specifying a range, it is up to the client how many resources it allocates. The default configuration when omitting this parameter is "0..infinity", i.e., no restriction.) • floating: The <i>amount</i> parameter refers to the number of licenses leased to a device. • total: The <i>amount</i> parameter refers to the total number of licenses on the connected device, floating licenses leased by this server plus licenses (pre-)installed on the device. <p>Note To allow a client to lease all type of licenses without restriction, enter the "resource any" command and remove all other resource commands. Or, if you want to restrict the license types leased to a device and don't want to lease other non-listed licenses, you have to list the resources explicitly and also add the "no resource any" command.</p>

Example: Create a default nodems-device profile that allows any type of licenses to be leased by clients. (The system already pre-creates this *DEFAULT* profile.).

```
profile nodems-device DEFAULT
resource any
```

Example: Create a nodems-device profile for a vSBC (Session-Border Controller running in a virtual machine). Each VM needs one "virtual-instance" license. In addition, we allow the VM to make up to 1000 sip-sessions, and we add a SIP-TLS-SRTP license. We don't want virtual clients to lease any other licenses, hence the "no resource any" command..

```
profile nodems-device VIRTUAL
resource virtual-instance
resource sip-sessions ..1000 total
resource sip-tls-srtp
no resource any
```

Link devices to a nodems-device profile

After having created a nodems-device profile we have to assign it to a connecting device. This is done by creating a *device* mode within an *organization* and *sub-organization* and by using the nodems-device profile within this device.

Procedure: Link a device to a nodems-device profile

Mode: NodeMS-Server / Organization / Sub-Organization

Command	Purpose
<code>node(sub-org)[sub-organization-name]# [no] device (serial-number any)</code>	Creates a device configuration entry and enters its mode. Use the serial-number to identify the device. If any is entered instead of the serial-number, this device configuration entry will be applied to all devices not listed explicitly. Note If no "device any" entry exists in a sub-organization, connections from devices that are not listed explicitly will be rejected. This allows the administrator of the License Server to whitelist the devices that are allowed to lease licenses from the server.
<code>node(device)[device-name]# [reset] use profile nodems-device profile-name</code>	Uses the specified nodems-device profile to determine how many licenses shall be leased to this device. The <i>DEFAULT</i> profile is used by default.

Example: Create a sub-organization for all virtual SBC instances and use the previously created *VIRTUAL* nodems-device profile.

```
nodems-server
    organization MY_ORG
        sub-organization VIRTUAL_SUBORG
        organization-key 0ABF8445ADF32D563FFE8D3564DCAB59
            device any
            use profile nodems-device VIRTUAL
nodems-server
no shutdown
```

Example: Set up a license-server for the configuration as depicted in [figure 13](#) on page 107:

```
profile nodems-device DEFAULT
resource any

nodems-server
    organization COMPANY1
        sub-organization TESTLAB
```

```

organization-key 00112233445566778899AABBCCDDEEFF
resource sip-sessions ..1000

device any
  use profile nodems-device DEFAULT

sub-organization PRODUCTIVE
  organization-key 0ABF8445ADF32D563FFE8D3564DCAB59
  resource sip-sessions ..5000

device any
  use profile nodems-device DEFAULT

organization COMPANY2

sub-organization DEFAULT
  organization-key 07FEA4829BCE3100AF39234AF982084E

device any
  use profile nodems-device DEFAULT

nodems-server
  no shutdown

```

Inspect the license-lease database

After having created a nodems-device profile we have to assign it to a connecting device. This is done by creating a *device* mode within an *organization* and *sub-organization* and by using the nodems-device profile within this device.

Procedure: Display License Server Status.

Mode: Operator Exec

Command	Purpose
<pre>> show nodems-server [detail detail-level] [continuously]</pre>	<p>Displays the status of the NodeMS-server and summarizes or lists all device connections.</p> <ul style="list-style-type: none"> detail-level: Determines how much information details are printed <ol style="list-style-type: none"> Shows server status and number of client connections (default detail level) Additionally, lists all client connections including the NodeMS services in use Additionally, more status information about the NodeMS services in use continuously: Reprints the output every second until Ctrl-C is pressed

Example: Show License Server status and number of connections.

```
>show nodems-server
```

```
NodeMS Server
=====
```

```
Configuration
-----
```

```
Listening Port:          443
Admin State:             up
```

```
Connections:             2
```

Example: Show License Server status and list all connections

```
>show nodems-server detail 2
```

```
NodeMS Server
=====
```

```
Configuration
-----
```

```
Listening Port:          443
Admin State:             up
```

```
Connections
-----
```

```
172.16.43.46:44743
-----
```

```
Protocol Phase:          Up
Session Identifier:      00A0BA09FAA2
Connected Since:         Thursday August 04 2016 12:09:49
Last Message Received:  6.764 s ago
Last Message Sent:      6.763 s ago
```

```
Services
-----
```

```
Service:                 Endpoint Authentication
Service:                 Session Establishment
Service:                 Remote command-line execution
Service:                 Resource Allocation
```

```
172.16.46.198:58638
-----
```

```
Protocol Phase:          Up
Session Identifier:      00A0BA09F412
Connected Since:         Friday August 05 2016 08:08:02
Last Message Received:  23.109 s ago
Last Message Sent:      23.108 s ago
```

```
Services
```

```

Service: Endpoint Authentication
Service: Session Establishment
Service: Remote command-line execution
Service: Resource Allocation

```

Procedure: Display devices connected to the License Server.

Mode: Operator Exec

Command	Purpose
> show nodems-server devices [organization organization-name] [sub-organization sub-organization-name] [device serial-number] [continuously]	Displays a list of all connected devices. <ul style="list-style-type: none"> • <i>organization-name</i>: Only displays devices within this organization. • <i>sub-organization-name</i>: Only displays devices within this sub-organization. • <i>serial-number</i>: Only displays devices with the specified serial-number. • continuously: Reprints the output every second until Ctrl-C is pressed

Example: List all devices connected to this License Server.

```

>show nodems-server devices

Sub-Organization      Device                Model                Online
-----
PATTON/TEST          00A0BA09F412         SN5300/4B2G          yes
PATTON/TEST          00A0BA09FAA2         SN5300/4B            yes

```

Procedure: Display on the License Server the geographical location of connected devices.

Mode: Operator Exec

Command	Purpose
> show nodems-server devices location [organization organization-name] [sub-organization sub-organization-name] [device serial-number] [continuously]	Displays a list of all connected devices and their geographical location. <ul style="list-style-type: none"> • <i>organization-name</i>: Only displays devices within this organization. • <i>sub-organization-name</i>: Only displays devices within this sub-organization. • <i>serial-number</i>: Only displays devices with the specified serial-number. • continuously: Reprints the output every second until Ctrl-C is pressed

Example: List the geo-location of all devices connected to this License Server.

```
>show nodems-server devices location
```

Sub-Organization	Device	Model	Online
PATTON/TEST	00A0BA09F412	SN5300/4B2G	yes >1

```
cont.
  Location Information
-----
>1 Device IP Address  111.111.111.111
   Autonomous System AS11111
   ISP                My ISP
   Organization       My Provider
   City               Nafada
   Region             Nafada
   Country            Nigeria
   Location           http://maps.google.com?q=11.11111,11.11111
```

Note Each device tries to determine its geo-location by sending an HTTP request to a public web service from its own global IP address. If the device is in a private network behind a NAT, the Device IP Address and the resulting geo-location will be that of the NAT server. Devices without a connection to the public internet are not able to determine their location at all.

Procedure: Display on the License Server the licenses leased by connected devices, sorted by the device serial-number.

Mode: Operator Exec

Command	Purpose
<pre>> show nodems-server devices licenses [organization organization-name] [sub-organization sub-organization-name] [device serial-number] [continuously]</pre>	<p>Displays a list of all connected devices and the licenses they currently lease.</p> <ul style="list-style-type: none"> • <i>organization-name</i>: Only displays devices within this organization. • <i>sub-organization-name</i>: Only displays devices within this sub-organization. • <i>serial-number</i>: Only displays devices with the specified serial-number. • continuously: Reprints the output every second until Ctrl-C is pressed

Example: List on the License Server the licenses currently leased by all devices connected to this License Server, sorted by the device-serial number.

```
>show nodems-server devices licenses
```

Sub-Organization	Device	Model	Online
PATTON/TEST	00A0BA09F412	SN5300/4B2G	yes >1

```

PATTON/TEST          00A0BA09FAA2          SN5300/4B          yes    >2
cont.
  License              Leased  Remaining
  -----
>1 sip-registrar      1      23h 59m 5s
   sip-sessions       188    23h 59m 5s
>2 sip-registrar      1      23h 59m 5s
   sip-sessions       188    23h 59m 5s

```

Procedure: Display on the License Server the licenses leased by connected devices, sorted by the device serial-number.

Mode: Operator Exec

Command	Purpose
> show nodems-server licenses [license-name] [organization organization-name] [sub-organization sub-organization-name] [device serial-number] [continuously]	Displays a list of all licenses and to which devices they are currently leased. <ul style="list-style-type: none"> • <i>license-name</i>: Only displays the specified license • <i>organization-name</i>: Only displays devices within this organization. • <i>sub-organization-name</i>: Only displays devices within this sub-organization. • <i>serial-number</i>: Only displays devices with the specified serial-number. • continuously: Reprints the output every second until Ctrl-C is pressed

Example: List the licenses currently leased by all devices connected to this License Server. Shows how the licenses are distributed across the different organizations, sub-organizations, and devices.

```

>show nodems-server licenses sip-sessions

License: sip-sessions
=====

Partition (Organizations, Device)          Limit   Leased  Remaining
-----
Total Floating                             10000
Total Floating Free                         9624
Total Floating Leased                       -        376

  Organization: PATTON                      -        376
    Sub-Organization: TEST                  -        376
      Device: any                           -        376
        Session: 00A0BA09F412              -        188   23h 59m 48s
          Session: 00A0BA09FAA2            -        188   23h 59m 48s
        Sub-Organization: VIRTUAL           -         -
          Device: any                       -         -

```

Configure Patton devices to lease licenses from a License Server

Instead of installing licenses permanently, Patton devices may lease floating licenses from a license server. The previous section explained how to set up a license server. This section describes how client devices (e.g. eSBCs, VoIP Gateways, IP Routers, etc.) are connected to the License server in order to lease licenses temporarily. Note that you can use both methods in parallel: install some licenses permanently on the device and lease the remaining licenses temporarily from a License Server.

In order to lease licenses temporarily the client device must be able to establish a TLS/TCP connection to the server. The device exhibits the following behavior if this connection is broken:

- If the client device shuts down or reboots normally, floating licenses are returned to the server's pool.
- If the client device loses contact to the server due to network problems, the leased licenses are still valid on the client for one day (or a shorter period configured on the client or server). If the device re-connects to the server, the leases are refreshed immediately. (During normal operation, they are refreshed every minute.)
- If the client device reboots unexpectedly the License Server keeps the leased licenses reserved for that device until it reconnects after boot-up (for a maximum period of one day). If the device does not come up again within one day, the server returns the licenses leased by that device back to its license pool and makes them available for other devices.

Note If you replace a client device with a new one, the License Server still keeps the licenses leased to the old device reserved for one day but immediately requires licenses for the new one. Make sure that your License Server is equipped with enough licenses for this scenario.

- If the License Server reboots unexpectedly, the client keeps the leased licenses for one day. If the License Server comes up again, the client device tries to refresh the lease immediately.

Prepare the system

Before you are able to connect a device to a License Server you have to prepare the system as follows:

1. Make sure the device is able to reach the License Server over the network, e.g. configure network interfaces and routing. If you have configured an ACL (Access-Control List), enable outgoing TCP connections to the server's port 443 (or the port configured in the *nodems-server* mode on the License Server) (see chapter 38, “[Access Control List Configuration](#)” on page 440)
2. Make sure the clock is accurate. The best way to grant this is to get the clock updates from an NTP server (see chapter 33, “[SNTP Client Configuration](#)” on page 390). This is required because the License Server authenticates itself with a self-generated certificate; this certificate is valid from the time you start the server. If the clock is inaccurate, client devices may refuse connecting to the server.

Connect a Patton device to a License Server

The following procedure shows how to configure a Patton device to lease licenses from a License Server.

Procedure: Connect a Patton device to a License Server and configure which licenses to lease..

Mode: Configure

Command	Purpose
<code>node(cfg)# nodems-client</code>	Enters the NodeMS-Client configuration mode. The license-distribution system is part of Patton's Node-Management System (NodeMS). In the future, more services will be added to manage Patton devices remotely over one secure communication channel.
<code>node(nodems-client) # [reset] server server-url</code>	Configures the URL used to connect to the License Server. The default server URL is <code>https://nodems.patton.io</code> . The following URLs are examples of valid URLs. Note that you can skip the <code>http://</code> protocol prefix and the default port: <ul style="list-style-type: none"> • <code>http://my-license-server</code> • <code>http://my-license-server:4443</code> • <code>10.10.1.1:4443</code> • <code>10.10.2.2</code>
<code>node(nodems-client) # [no] organization-key key</code>	Configures the secret shared with the License Server's sub-organization configuration. <ul style="list-style-type: none"> • key: The same key must be configured in the corresponding sub-organization mode on the License Server.
<code>node(nodems-client) # [reset] allocate (auto lease for amount (minutes hours days weeks) permanent)</code>	Configures whether the received licenses are installed persistently (not supported yet) or leased temporarily and for how long. <ul style="list-style-type: none"> • lease for: Licenses are leased for the specified period of time. If the device loses contact to the License Server it is able to continue using the leased license for this period of time. • permanent: Licenses are permanently installed (not supported yet). • auto: Licenses are leased for one day. This is the default configuration.

Command	Purpose
<p>node(pf-ralloc)[profile-name]# [no] resource resource-name [amount] [floating total]</p> <p>node(pf-ralloc)[profile-name]# [no] resource any</p>	<p>Specifies how many licenses this device wants to lease from the License Server. If the no-form of the command is used, the device does not request the specified resource.</p> <ul style="list-style-type: none"> • resource-name: Name of the license (e.g. <i>sip-sessions</i>). any is a placeholder for all licenses not specified explicitly. • amount: Number of licenses this client wants to lease. You can either specify a number (e.g. "10") or a number range (e.g. "..10" for between zero to ten, e.g. "10..100"). When specifying a range, it is up to the server how many resources it wants to allocate for this device. The default configuration when omitting this parameter is "0..infinity", i.e., no restriction. In this case the device gets all available licenses up to its nominal capacity. • floating: The amount parameter refers to the number of licenses leased from the server. • total: The amount parameter refers to the total number of licenses on this device, floating licenses leased from the server plus licenses (pre-)installed on the device. <p>Note By default, the "resource any" command line allows the device to receive licenses the server may lease. If you want to restrict the received licenses, enter the "no resource any" command.</p>

Display information about the License Server connection

The following commands can be executed on the client to show how whether the client is successfully connected to a License Server.

Procedure: Display status and connection information.

Mode: Operator Exec

Command	Purpose
<pre>> show nodems-client [detail detail-level] [continuously]</pre>	<p>Displays the status of the NodeMS-client</p> <ul style="list-style-type: none"> detail-level: Determines how much information details are printed <ol style="list-style-type: none"> Shows server status and number of client connections (default detail level) Additionally, lists all client connections including the NodeMS services Additionally, more status information about the NodeMS services continuously: Reprints the output every second until Ctrl-C is pressed

Example: Show License Client status.

```
>show nodems-client

NodeMS Client
=====

Configuration
-----

Server:                172.16.46.46:443
Organization:          PATTON
Sub-Organization:      TEST
Organization-Key:      (set)
Admin State:           up

Status
-----

Connection State:      Connected
Protocol Phase:        Up
Session Identifier:    00A0BA09F412
Connected Since:       Friday August 05 2016 18:02:28
```


Procedure: Display licenses leased from the License Server.

Mode: Operator Exec

Command	Purpose
> show nodems-client licenses [continuously]	Displays the licenses leased from the License Server <ul style="list-style-type: none"> • continuously: Reprints the output every second until Ctrl-C is pressed

Example: Show licenses leased from the License Server.

```
>show nodems-client licenses
```

```
Resource                Limit      Leased  Remaining
-----
iprouter                 ..1 total      -
packetSMART              ..1 total      -
rootkey                  ..1 total      -
sip-registrar            ..1 total       1  23h 58m 56s
sip-sessions              ..192 total    188  23h 58m 56s
sip-tls-srtp             ..1 total      -
transcoding              ..1 total      -
voice-channels           -              -
vpn                       ..1 total      -
```

Chapter 9 AAA Configuration

Chapter contents

Introduction	127
The AAA Component	127
General AAA Configuration	128
Configuring TACACS+ client	129
Configuring TACACS+ server	129
Authentication	129
Authorization	129
Server configuration example	130
Configuring RADIUS clients	132
Configuring RADIUS server	132
Attributes in the RADIUS Authentication request message	132
Attributes in RADIUS Authentication request for Authorization	132
Attributes in RADIUS Accept message	133

Introduction

This chapter provides an overview of the AAA (Authentication, Authorization, and Accounting) component and describes how to configure the TACACS+ or RADIUS clients, which are subparts of the AAA component. It is important to understand how AAA works before configuring the TACACS+ or RADIUS clients. This chapter also describes the local database accounts configuration, which is another subpart of AAA.

To use the authentication and authorization service on Trinity, you have to configure the AAA component, the Radius component, the TACACS+ component and the local database accounts.

This chapter includes the following sections:

- The AAA component
- TACACS+ configuration
- RADIUS component
- Configuration of the local database accounts

The AAA Component

Authentication, authorization, and accounting is a term for controlling access to client resources, enforcing policies, auditing usage, and providing information necessary to invoice users for services.

Authentication provides a way of identifying a user (usually in the form of a login window where the user is expected to enter a username and password) before allowing access to a client. The AAA component compares the user's authentication login information with credentials stored in a database. If the information is verified, the user is granted access to the network. Otherwise, authentication fails and network access is denied.

Following authentication, authorization determines the activities, resources, or services a user is permitted to access. For example, after logging into a system, a user may try to issue commands, the authorization process determines whether the user has the authority to issue such commands.

Accounting, which keeps track of the resources a user consumes while connected to the client, can tally the amount of system time used or the amount of data transferred during a user's session. The accounting process records session statistics and usage information that is used for authorization control, billing, and monitoring resource utilization.

AAA information can be stored in a local database or in a database on a remote server. A current standard by which network access servers interface with the AAA server is the TACACS+ Service (Terminal Access Controller Access-Control System Plus) or RADIUS (Remote Authentication Dial In User Service).

[Figure 14](#) on page 128 illustrates the authentication procedure for a user logging into a SmartNode that is configured to use RADIUS as authentication method.

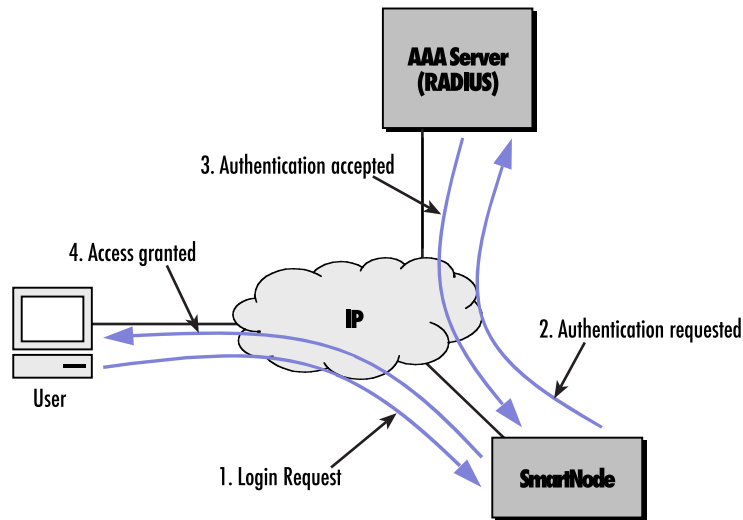


Figure 14. Authentication procedure with a TACACS+ server

General AAA Configuration

The AAA component consists of AAA profiles and AAA methods. A service (e.g. Telnet) has to specify a profile it wants to apply to all login requests. The profile then specifies the sequence in which methods are applied to obtain AAA information. Figure 15 illustrates the correlation between the Telnet login and console login services.

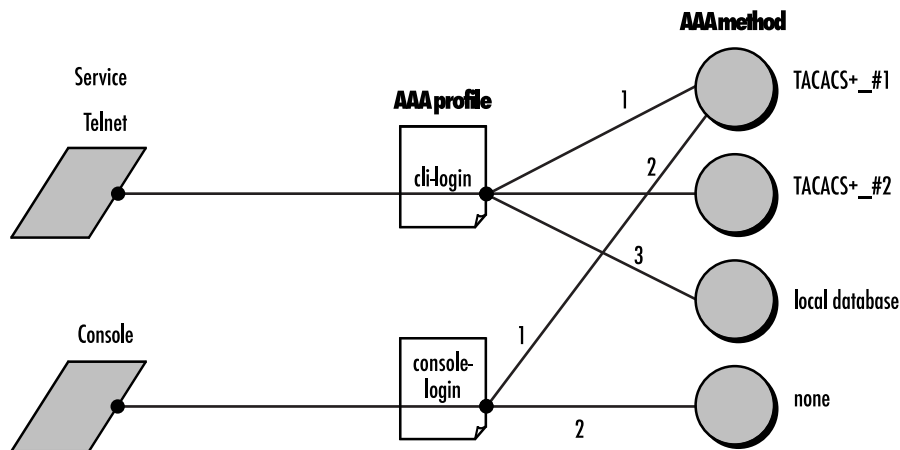


Figure 15. How to use AAA methods and AAA profiles

The Telnet service uses an AAA profile called *cli-login*. This profile specifies that the following methods are used in the order they appear in the configuration:

1. Query TACACS+ server TACACS+_#1.
2. Query TACACS+ server TACACS+_#2.

- Query the local database (see the "Configuring operators, administrators, and superusers" section for information on how to configure the local database).

If, e.g. TACACS+_#1 is not available, TACACS+_#2 will be queried after a timeout. But if TACACS+_#1 gives an answer that rejects the login request, the remaining methods are not used and the login is denied. The same applies to the console service, which uses the profile console-login. This profile uses the following sequence of methods:

- Ask TACACS+ server TACACS+_#1.
- Ask predefined method none. This method always grants access as system operator.

If TACACS+_#1 is not available, access will be granted by the method none. If TACACS+_#1 rejects the login request, console access is denied. If TACACS+_#1 confirms the request, console access is granted.

Configuring TACACS+ client

If the AAA profiles you have defined make use of the TACACS+ AAA method, you must configure the corresponding TACACS+ clients. To configure TACACS+ client, do the following steps:

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg)# tacacsplus-client <name></code>	Adds a TACACS+ client with the name <i>ame</i> and enters TACACS+-client configuration mode.
2	<code>node(tacplus)[device]# server host-name> [<port>]</code>	Sets the host name (or IP address) of the remote TACACS+ server. If no port is specified then the default port 49 is automatically set.
3	<code>node(tacplus)[device]# [no] shared-secret <key></code>	Sets the password shared between the TACACS+ client and the remote TACACS+ server. When no password is set, then an empty password is sent during authentication.

Configuring TACACS+ server

Authentication

During authentication process the client send a standard ASCII authentication request to the server containing the user name and password. Other authentication type (PAP, CHAP, ARAP and MSCHAP) are not supported by the client. Users with password should be configured on the server using the standard ASCII authentication. An authentication request is usually followed by an Authorization request.

Authorization

Authorization request are usually sent after an authentication request and are used to ask the server if the user is allowed to use the requested service. An authorization request always contains the requested service. The client supports only the following services:

Service	Description
telnet	User telnet login request.
ssh	User ssh login request.
fcgi	User webpage login request.
console	User console login request.
call	User call request.

For authorization login request (telnet, ssh, fcgi and console) the server must response with the following attribute-value pair:

Attribute Name	Attribute Value	Description
patton-priv-lvl	operator	The user as operator privilege
	administrator	The user as administrator privilege
	superuser	The user as superuser privilege

For call authorization request, the user field in the request is filled with the e-164 called number. The following attribute-value pairs are added to the request if they are available:

Attribute Name	Attribute Value
nas-identifier	<id>
calling-station-id	<id>
called-station-id	<id>
calling-ip-address	<ip-address>
called-ip-address	<ip-address>

All attributes present on the request must be configured on the server to allow a call! The server can repeat an attribute-value pair, to allow more than one call ID/IP.

Server configuration example

Below is an example of configuration file for the TACACS+ F4.0.4.10 Linux server:

```
# tacacs+ configuration file
# /etc/tac_plus.conf

# set the shared secret key between client and server
key = cle_tacacs

# set the accounting file, where all accounting request are logged
accounting file = /var/log/tac_plus.acct

# users accounts

user = johndoe {
  login = cleartext "normal"      # enter login password in clear text
  name = "John Doe"              # Name description (not used by trinity)

  # You must now list all service that the user is allowed to connect with
```

```

# ( console | telnet | ssh | fcgi ):

service = telnet {
    patton-priv-lvl = operator # set the privilege level of the
                                # user (operator|administrator|superuser)
}

service = ssh {
    patton-priv-lvl = administrator # set the privilege level of the user
                                    # (operator|administrator|superuser)
}

# Example with encrypted password

user = johndoe {
    login = des "yrVMIa532Sy.2" # enter login password encrypted with
                                # tac_pwd program
name= "John Doe"

    service = telnet {
        patton-priv-lvl = administrator # set the privilege level of the user
                                        # (operator|administrator|superuser)
    }
    service = ssh {
        patton-priv-lvl = superuser # set the privilege level of the user
                                    # (operator|administrator|superuser)
    }
}

# call accounts to authorize call

user = 100 { # user is the e-164 called number
    service = call {
        nas-identifier = MyNas
        calling-station-id = 100
        called-station-id = 200 # Allow to call 200 and 300
        called-station-id = 300
        calling-ip-address = 192.168.0.1
        called-ip-address = 192.168.0.2 # Allow called IP address 192.168.0.2 and
192.168.0.3
        called-ip-address = 192.168.0.3
    }
}

```

Configuring RADIUS clients

If the AAA profiles you have defined make use of the RADIUS AAA method, you must configure the corresponding RADIUS clients. To configure RADIUS clients, do the following steps:

Mode: Configure

Step	Command	Purpose
1	node(cfg)#radius-client <name>	Adds a RADIUS client with name <i>name</i> and enters RADIUS-client configuration mode.
2	node(radius)[name]#server [auth-port <port>] [acct-port <port>]	Sets the hostname (or IP address) of the remote RADIUS server. Optionally the Authentication Port (default: 1812) and Accounting Port (default: 1813) can be given.
3	node(radius)[name]#shared-secret <key>	Sets the password shared between the RADIUS client and the remote RADIUS server.
4	node(radius)[name]#show aaa radius	Displays configured RADIUS servers.

Configuring RADIUS server

Each message to and from a RADIUS server includes several attributes. Attributes are, For example, in a login request, the name and password of the user that requires to login. For more information about each attribute, or other possible attributes, see RFC 2865 or the documentation of the radius server you use.

Attributes in the RADIUS Authentication request message

Patton device sends a RADIUS Authentication request with the following attributes:

Attribute number	Attribute type	Description
1	User-Name	Indicates the name of the user to be authenticated
2	User-Password	Indicates the password of the user to be authenticated
6	Service-Type	For Authentication it has always the following value: 1. LOGIN - Authorize user to login
61	NAS-Port-Type	The NAS port type is always set to VIRTUAL (value = 5)

Because RADIUS doesn't support Authorization, the Patton device sends a second RADIUS Authentication request for Authorization. There are two different cases for Authorization:

1. Login Authorization

Attributes in RADIUS Authentication request for Authorization

The following attributes are sent in a RADIUS Authentication Request for User login Authorization:

Attribute number	Attribute name	Attribute type	Description
1	User-Name	string	Indicates the name of the user or call to be authorized

Attribute number	Attribute name	Attribute type	Description
6	Service-Type	integer	It has one of the following values: 1. LOGIN - Authorize user to login
61	NAS-Port-Type	integer	The NAS port type is always set to VIRTUAL (value = 5)
Patton Vendor Attributes			
26	Patton-Protocol	string	The value can be one of the following: <ul style="list-style-type: none"> • telnet • ssh • console • http • snmp

Attributes in RADIUS Accept message

After the user and his credentials are approved by the authentication procedure on the RADIUS server, the Patton device expects a RADIUS accept message with the following attributes:

Attribute number	Attribute name	Attribute type	Description		
18	Reply-Message	string	Contains the text that is printed to the user after login. If the attribute is not included in the message, no text will be printed.		
6	Service-Type	integer	Optional attribute to set the user rights. Note that if the Vendor Patton-Group attribute is present, the Service-Type is ignored.		
			Value	Name	User right
			6	Administrative	superuser
			7	NAS-Prompt	operator
Patton Vendor Attributes					
17	Patton-Group	string	Defines the user rights, it can have the following values: <ul style="list-style-type: none"> • superuser • administrator • operator If the attribute is missing, then the user has superuser rights by default.		
18	Patton-Web-Interface-Type	string	Defines the web interface type: <ul style="list-style-type: none"> • ADVANCED • BASIC By default ADVANCED web interface is granted.		

2. Call Authorization

Attribute number	Attribute name	Attribute type	Description
1	User-Name	string	Indicates the call to be authorized Indicates the password of the call to be authorized
2	User-Password	string	It has one of the following values: 10. CALL-CHECK - Authorize call
6	Service-Type	integer	The NAS port type is always set to VIRTUAL (value = 5)
32	NAS-Identifier	string	NAS Identifier
61	NAS-Port-Type	integer	The NAS port type is always set to VIRTUAL (value = 5)
44	Acct-Session-Id	string	
Patton Vendor Attributes			
85	Patton-Calling-Station-Id	string	Identifier of the calling station (usually E.164 number)
53	Patton-Called-Station-Id	string	Identifier of the called station (usually E.164 number)

Note Call doesn't send Authentication request but only Authorization request, which means that for RADIUS only one Authentication request is sent for call while two Authentication requests are sent for user login.

Chapter 10 **Basic System Management**

Chapter contents

Introduction	136
Basic System Management Configuration Task List	136
Managing Feature License Keys	137
Showing System Resources	138
Setting System Parameters	138
Setting the System Banner	140
Setting Time and Date	140
Configuring Daylight Savings Time Rules	141
Display Clock Information	142
Display Time Since Last Restart	142
Configuring and starting the web server	142
Restarting the system	143
Displaying the System Logs	143
Displaying the System Logs	144
Exporting System Logs and Reports	145
Configuring the blink interval	146
Configuring the Syslog Client	146
Factory Reset	147
Reset Button	147
Trinity Performance Tracker	147

Introduction

This chapter describes parameters that report basic system information to the operator or administrator, and their configuration. The following are basic parameters that can be established when setting up a new system (see section “[Setting System Parameters](#)” on page 138):

- Defining the system's hostname
- Setting the location of the system
- Providing reference contact information
- Setting the clock

Additionally, the following tasks are described in this chapter:

- Setting the system banner (see section “[Setting the System Banner](#)” on page 140)
- Enabling the embedded web server (see section “[Configuring and starting the web server](#)” on page 142)
- Using the Trinity Performance Tracker (see section “[Trinity Performance Tracker](#)” on page 147)

Basic System Management Configuration Task List

All tasks in the following sections are optional, though some such as setting time and calendar services and system information are highly recommended.

To configure basic system parameters, perform the tasks described in the following sections.

- Managing feature license keys (see [page 137](#))
- Setting system information (see [page 138](#))
- Setting the system banner (see [page 140](#))
- Setting time and date (see [page 140](#))
- Displaying clock information (see [page 142](#))
- Displaying time since last restart (see [page 142](#))
- Configuring and starting the secure web server (see [page 142](#))
- Restarting the system (see [page 143](#))
- Displaying the System Logs (see [page 143](#))
- Displaying Reports
- Exporting System Logs and Reports
- Identifying a unit by flashing all LED's (see [page 145](#))

Managing Feature License Keys

Several features of the firmware require a system specific license key to be installed to enable the feature.

This section describes how to install the feature license keys on your equipment.

Mode: Configure

Step	Command	Purpose
1	device(cfg)#install license <i>license-key</i>	Install the license key
2		Repeat step 1 for any additional license keys

Example: Installing license keys from the console

The following example shows the command used to install license keys manually on the console.

```
device(cfg)#install license 10011002R1Ws63yKV5v28eVmhDsVGj/JwKqIdpC4Wr1BHaN-
tenXUYF/2gNLoihifacaTPLKcV+uQDG8LJis6EdW6uNk/GxVObDEwPFJ5bTV3bIIIfUZ1eUe+8c5OpC-
Cd7PSAe83Ty2c/
CnZPS1eJiRv1Jrr8VhOr1DYxkEV9evBp+tSY+y9sCeXhDWT5Xq15SAP1znTLQmym7fDakvm+zltzswX/
KX13sdkR0ub9IX4Sjn6YrvkyrJ2dCGivTTB3iOBmRjV1u
```

After installing license keys, you can check if the license keys have been added successfully to your system using the following command.

Mode: Configure

Step	Command	Purpose
1	device(cfg)#show system licenses	Displays all dynamic and static licenses

Example: Displaying installed licenses

The following example shows the command used to display all installed licenses on a system and a sample of its output.

```
device(cfg)#show system licenses
Dynamic Licenses
=====

Name: sip-tls-srtp
-----

ID:                               54
Description:                       SIP TLS and SRTP

Name: sip-registrar
-----

ID:                               55
Description:                       SIP Registrar

Static Licenses
```

Showing System Resources

The following command will display all available resources including Voice Ports and total number of SIP Legs. The system information includes the following parameters:

- Contact
- Hostname
- Location
- Provider
- Subscriber
- Supplier

Mode: all

Step	Command	Purpose
1	node#show system resources	Displays available resources including Voice Ports and total number of SIP Legs

Example:

```
(cfg)#show system resources
Ports
=====

BRI Ports:                0
E1T1 Ports:              4
FXS Ports:                0
FXO Ports:                0

VoIP Resource
=====

Number of DSPs:           2
DSP Firmware:             0
DSP Channels:             120
SIP Legs:                 180
Transcoding:              no
Border Controller:        no
```

Setting System Parameters

The system information includes the following parameters:

Note By default there is no information specified for any of the following parameters.

- **Contact:** System contact information tells the user how to contact the information service, e.g. the help line of the service provider. The contact information may be any alphanumeric string, including spaces, that is no longer than one line. This entry corresponds to the MIB II system sysContact object.

- **Hostname:** The system name, also called the hostname, is used to uniquely identify the Patton device in your network. The selected name should follow the rules for ARPANET hostnames. Names must start with a letter, end with a letter or digit, and have as interior characters only letters, digits, and hyphens. Names must be 63 characters or fewer. For more information, refer to RFC 1035. This entry corresponds to the MIB II system sysName object. After setting the hostname of the Patton device the CLI prompt will be replaced with the chosen name.
- **Location:** Assigning explanatory location information to describe the system physical location of your device (e.g. server room, wiring closet, 3rd floor, etc.) is very supportive. This entry corresponds to the MIB II system sysLocation object.
- **Provider:** The system provider information is used to identify the provider contact for this Patton device, together with information on how to contact this provider. The provider is a company making services available to subscribers. The provider information may be any alphanumeric string, including spaces, that is no longer than one line. This entry corresponds to the Patton Electronics enterprise-specific MIB provider object.
- **Subscriber:** The system subscriber information is used to get in touch with subscriber for this Patton device, together with information on how to contact this subscriber. The subscriber is a company or person using one or more services from a provider. The subscriber information may be any alphanumeric string, including spaces, that is no longer than one line. This entry corresponds to the Patton Electronics enterprise-specific MIB subscriber object.
- **Supplier:** The system supplier information is used to get in touch with the supplier for this Patton device, together with information on how to contact this supplier. The supplier is a company delivering Patton devices to a provider. The supplier information may be any alphanumeric string, including spaces, that is no longer than one line. This entry corresponds to the Patton Electronics enterprise-specific MIB supplier object.

Mode: Configure

Step	Command	Purpose
1	<code>device(cfg)#system contact information</code>	Sets the contact information to <i>information</i>
2	<code>device(cfg)#system hostname information</code>	Sets the hostname to <i>information</i>
3	<code>device(cfg)#system location information</code>	Sets the location information to <i>information</i>
4	<code>device(cfg)#system provider information</code>	Sets the provider information to <i>information</i>
5	<code>device(cfg)#system subscriber information</code>	Sets the subscriber information to <i>information</i>
6	<code>device(cfg)#system supplier information</code>	Sets the supplier information to <i>information</i>

Note If the system information must have more than one word, enclose it in double quotes.

Example: Setting system information

The following example shows the commands used to configure the contact information for your device, if you start from the operator execution mode.

```
device(cfg)#system contact "Bill Anybody, Phone 818 700 1504"
device(cfg)#system hostname device
device(cfg)#system location "Wiring Closet, 3rd Floor"
```

```
device(cfg)#system provider "Best Internet Services, contact@bis.com, Phone 818 700
2340"
device(cfg)# system subscriber "Mechanical Tools Inc., jsmith@mechtool.com, Phone
818 700 1402"
device(cfg)# system supplier "WhiteBox Networks Inc., contact@whitebox.com, Phone
818 700 1212"
```

Setting the System Banner

The system banner is displayed on all systems that connect to your Patton device via Telnet, SSH, or a serial connection. It appears at login and is useful for sending messages that affect administrators and operators, such as scheduled maintenance or system shutdowns. By default no banner is present on login.

To create a system banner use the **banner** command followed by the message you want displayed. If the banner message has to be formed out of more than one word the information is enclosed by double quotes. Adding the escape sequence "\n" to the string forming the banner creates a new line on the connected terminal screen. Use the **no banner** command to delete the message.

```
Mechanical Tools Inc.
jsmith@mechtool.com
Phone 818 700 1402
```

```
login:
```

Mode: Configure

Step	Command	Purpose
1	device(cfg)#banner <i>message</i>	Sets the message for the system banner to <i>message</i>

Example: Setting the system banner

The following example shows how to set a message for the system banner for your device, if you start from the configuration mode.

```
device(cfg)#banner \n#\n# The password of all operators has changed\n# please con-
tact the administrator\n#"
```

Setting Time and Date

All Patton devices provide time-of-day and date services. These services allow the products to accurately keep track of the current time and date. The system clock specifies year, month, day, hour, minutes, and optionally seconds. The time is in 24-hour format *yyyy-mm-ddThh:mm:ss* and is retained after a reload.

Mode: Configure

Step	Command	Purpose
1	device(cfg)#clock set <i>yyyy-mm-ddThh:mm:ss</i>	Sets the system clock to <i>yyyy-mm-ddThh:mm:ss</i>

Note The integrated SNTP client allows synchronization of time-of-day and date to a reference time server. Refer to chapter 33, “SNTP Client Configuration” on page 390 for more details.

Example: Setting time and date

The following example shows the commands used to set the system clock of your device to August 6, 2001 at 16:55:57, if you start from the operator execution mode.

```
device(cfg)#clock set 2001-08-06T16:55:57
```

Configuring Daylight Savings Time Rules

Trinity allows configuring daylight saving time rules, which affect the local clock offset without changing the configuration. After booting up and loading the configuration, the daylight saving rules are checked and applied automatically. The rules consist of a default-offset and one or multiple dst-rules. The offset of a dst-rule is active if the local clock is between the specified start and stop time of the rule. If the local clock is outside the specified start and stop time of all specified rules, then the default-offset is active.

Note When the DST rule is active, its offset is added to the local default offset. Therefore, the DST offset defines how much time is “shifted” during summer (e.g. 1h).

Mode: Configure

Step	Command	Purpose
1	device(cfg)#clock local default-offset (+hh:mm -hh:mm)	Configures the offset of your time zone from GMT. This offset is used if no other dst rule is currently active. Default: +00:00

Mode: Configure

Step	Command	Purpose
1	node(cfg)]#clock local dst-rule <name> (+hh:mm -hh:mm) from <month> <counter> <day-of-week> hh:mm [<year>] until <month> <counter> <day-of-week> hh:mm [<year>]	Configures a DST rule that enables summer time on a specific day of the week (e.g. last Sunday of March)

Mode: Configure

Step	Command	Purpose
2	node(cfg)]#clock local dst-rule <name> (+hh:mm -hh:mm) from <month> <day-of-month> hh:mm [<year>] until <month> <day-of-month> hh:mm [<year>]	Configures a DST rule that enables summer time on a specific day of the month (e.g. 23rd of September)

Possible parameters for the daylight savings time configuration commands:

Parameter	Possible Values
<month>	(jan feb mar apr may jun jul aug sep oct nov dec)
<day-of-week>	(monday tuesday wednesday thursday friday saturday sunday)

Parameter	Possible Values
<counter>	(first second third fourth last 1st 2nd 3rd 4th 5th 6h 7th 8th 9th 10th 11st 12nd 13rd 14th 15h 16th 17th 18th 19th 20th 21st 22nd 23rd 24th 25h 26th 27th 28th 29th 30th 31st

Display Clock Information

This procedure describes how to display the current date and time

Mode: Both in operator and administrator execution

Step	Command	Purpose
1	device>show clock	Display the local time.

Example: Display clock information

The following example shows the commands used to display the time and date settings of your device in local time, if you start from the operator execution mode.

```
device>show clock
2001-08-06T16:55:57
```

Display Time Since Last Restart

This procedure describes how to display the time since last restart

Mode: Operator execution

Step	Command	Purpose
1	device>show uptime	Display the time since last restart.

Example:

The following example shows how to display the uptime of your device, if you start from the configuration mode.

```
device>show uptime
The system is up for 54 days, 23 hours, 44 minutes, 18 seconds
```

Configuring and starting the web server

The embedded web server has multiple parameters that are configurable. HTTPS is now supported using a default SSL certificate that is unique to each device.

Note The web browser will warn when it connects to the web site. This is because the default SSL certificate is self-signed and is not assigned to a specific host-name, so the browser cannot use it to authenticate (i.e. to ensure that it is connected to the web site it thinks it is). Nevertheless, the connection will be encrypted.

Mode: Configure

Step	Command	Purpose
1	node(cfg)# web-server	Enters the web-server configuration mode
2	node(cfg)# [no] shutdown	Enables the web-server (default disabled).
3 (optional)	node(cfg)# [no] protocol http [port <port>]	Configures the http protocol (default enabled), and optionally sets the port number (default 80).
4 (optional)	node(cfg)# [no] protocol https [port <port>]	Configures the https protocol (default enabled), and optionally sets the port number (default 443).
5 (optional)	node(cfg)# display {advanced basic}	Sets the web interface display type.
6 (optional)	node(cfg)# webrefresh {5 30 60 300}	Sets how often in seconds the web interface should contact the device to refresh (default 30).
7 (optional)	node(cfg)# use profile aaa <profile>	Sets which AAA profile the web-server should use.

Restarting the system

In case the Patton device has to be restarted, the **reload** command must be used. The reload command includes a two-dialog, where the user is allowed to store any unsaved configuration data and finally confirms the system restart.



Restarting the system interrupts running data transfers and all voice calls.

Mode: Administrator execution

Step	Command	Purpose
1	device#reload	Restarts the system

The execution command **reload** has been enhanced with the following options:

- cancel** - Cancels reload/halt
- forced** - Reloads the system without prompting for confirmation.
- if-needed** - Only reloads if the system knows that a reload is needed.
- in <seconds>** - Sets the time in which to perform the reload/halt action.

The following example shows how to restart the running system, if you start from the administrator execution mode.

```
device#reload
Type 'yes' to restart/halt, anything else to cancel:
```

Displaying the System Logs

The system logs contain warnings and information from the system components of Trinity3. In case of problems it is often useful to check the event or the supervisor logs for information about malfunctioning system

components. The event log stores general events such as flash full, DSP failed etc., comparable with the event log on Windows NT. The supervisor log stores information from the system supervisor such as memory full, task failed etc.

System resets may have a number of reasons, the most prominent being a manual reset issued on the Telnet/console ('reload'). Other reset reasons include power off failures and system failures. In order to pinpoint the problem, the reset log contains the reset cause.

The show log command offers a new argument to suppress color information to be printed. Since Trinity 3.7 some log messages are printed in color, e.g. error messages in red. If log dumps are copied into text files, some editors have problems to render the color codes (ESC sequence) correctly. The 'unformatted' parameter removes these color codes before printing the log.

Mode: Operator execution

Step	Command	Purpose
1	node>show log boot [debug] [unformatted]	Displays the console and log messages captured during startup of the unit; the 'debug' option print more details.
2	node>show log error [unformatted]	Displays all error messages.
3	node>show log event [unformatted]	Shows important events such as link up/down.
4	node>show log file-transfer [unformatted]	Displays provisioned file transfers.
5	node>show log performance {hourly daily monthly}	Displays performance statistics of the device within the specified range.
6	node>show log reset [unformatted]	Outputs a list of reset reasons (with date and time).
6	node>show log supervisor [unformatted]	Shows a dump of the system supervisor, used, for example, to get information about an unexpected reboot.

Displaying the System Logs

The show reports command provides the 'unformatted' argument as well to suppress color codes in the output. This command is used to dump combined system information. We also changed the order in which the following logs are concatenated:

```
show log reset [unformatted]
show log error [unformatted]
show log event [unformatted]
show log file-transfer [unformatted]
show log supervisor [unformatted]
show log boot debug [unformatted]
show running-config
```

Mode: Operator execution

Step	Command	Purpose
1	<code>node>show reports [unformatted]</code>	Dumps the combined system information with (default) or without color highlighting ('unformatted' parameter).

Exporting System Logs and Reports

The copy command now contains a new "log:" source which allows for exporting all logs and reports to an external TFTP server. By default the log files are exported without color information, but you can include color with the optional 'formatted' argument.

Mode: Operator execution

Step	Command	Purpose
1	<code>node>copy log:boot tftp://<server-ip>/<path> [formatted]</code>	Copies the console and log messages captured during startup of the unit to text <i>filepath</i> on TFTP server <i>server-ip</i> .
2	<code>node>copy log:boot-debug tftp://<server-ip>/<path> [formatted]</code>	Copies the detailed console and log messages captured during startup of the unit to text <i>filepath</i> on TFTP server <i>server-ip</i> .
3	<code>node>copy log:error tftp://<server-ip>/<path> [formatted]</code>	Copies all error messages to text file <i>path</i> on TFTP server <i>server-ip</i> .
4	<code>node>copy log:event tftp://<server-ip>/<path> [formatted]</code>	Copies all important events such as link up/down to text file <i>path</i> on TFTP server <i>server-ip</i> .
5	<code>node>copy log:file-transfer tftp://<server-ip>/<path> [formatted]</code>	Copies performance statistics of the device to tape archive <i>filepath</i> on TFTP server <i>server-ip</i> .
6	<code>node>copy log:performance.tar tftp://<server-ip>/<path></code>	Copies performance statistics of the device to tape archive <i>filepath</i> on TFTP server <i>server-ip</i> .
7	<code>node>copy log:reports tftp://<server-ip>/<path> [formatted]</code>	Copies combined system information to text file <i>path</i> on TFTP server <i>server-ip</i> .
8	<code>node>copy log:reports.tar tftp://<server-ip>/<path> [formatted]</code>	Copies combined system information to tape archive <i>filepath</i> on TFTP server <i>server-ip</i> . The 'formatted' argument specifies whether the log files within the generated tape archive contain color codes. This method (without the 'formatted' argument) is the preferred file format to be sent to Patton support.
9	<code>node>copy log:reset tftp://<server-ip>/<path> [formatted]</code>	Copies a list of reset reasons to text file <i>path</i> on TFTP server <i>server-ip</i> .
10	<code>node>copy log:supervisor tftp://<server-ip>/<path> [formatted]</code>	Copies a dump of the system supervisor to text file <i>path</i> on TFTP server <i>server-ip</i> .

Configuring the blink interval

When there are many Trinity devices in the same location, use this command to flash all the LED's on a specific unit for a specified period of time. This makes identification of the physical unit very easy.

Step	Command	Purpose
1	device #blink <seconds>	Enter an integer for the period of time you want the LED's to flash on the physical unit.

Configuring the Syslog Client

Syslog is a protocol for sending event notification messages across IP networks to message collectors (Syslog server). It uses transport protocol UDP on port 514. A syslog-message exits on the three main part Priority, Header and Message whereas the header is split into Facility and Severity and the header into Timestamp and Hostname. The whole syslog-message (Priority, Header and Message) contains only printable characters and the maximum length is 1024 bytes.

Mode: Configure

Step	Command	Purpose
1	device(cfg)#syslog-client	Enters syslog client configuration mode.

Mode: Syslog Client

Step	Command	Purpose
1	device(syslog-client)#[no] remote { <ipv4 host> <ipv6 host> } [tcp udp] [<port>]	Creates a new remote destination and enters its configuration mode. The 'no' form of the command removes an existing remote destination. The protocol type and port are optional. If not included, the default UDP port 512 will be used.

Mode: Remote

Step	Command	Purpose
1	device(syslog-client)(remote)#[no] facility <service name> <severity>	Creates a new log expression for a remote destination. It exists on a facility that determines from which source messages must be accepted and a severity that defines up to which level the messages of the given facility must be sent. The 'no' form of the command disables sending of messages from the given facility.

Factory Reset

This command performs the same action as the reset button. It will currently remove the configurations, the logs, the preferences files and the installed TLS keys and certificates present in the flash.

Mode: administrator execution

Step	Command	Purpose
1	device# factory-reset	Reset to factory state

Reset Button

The reset button can be used as follows:

1. Hold the reset button during boot
2. The power LED flashes quickly for 2 seconds, during which time the reset button must remain pressed
3. The power LED will begin a blink pattern (described below)
4. Pressing the reset button will change the blink pattern
5. 10 seconds after the last reset button press, an action will be performed based on the selected pattern

Pattern	Action
1-blink, pause	Boot normally
2-blink, pause	Switch to backup image, then boot (Boot normally if the device only has a single image).
3-blink, pause	Erase all configuration and licenses, then boot.

Trinity Performance Tracker

The performance tracker is a new module of the system supervisor that continuously observes performance measures of the system such as CPU load, memory consumption, IP data traffic, etc. These samples help our support team to understand what load a device was confronted with when facing a problem. To investigate some dynamic issues you will be asked to provide these performance measures to our support team. This is done by uploading the measures as tape archive (tar-file) to your TFTP server with the copy command and send us the tar file by email.

Note The collected data **DO NOT** contain any confidential information such as IP addresses, call records, passwords, etc. If you are interested in the information exchanged with our support team you may open the tar file: You will find three CSV files (comma-separated values), in which each row contains the data of one measurement. The first rows provides information about the columns. The three files correspond to measures collected over the last hour, day, and month, respectively. Next to these CSV files there is a text file with meta-information about the device such as model and host name, serial number, etc.

As mentioned before performance measures are stored in RAM and are therefore lost when a power cycle occurs. However, the storage area survives a soft reboot, for example when manually reloading the device or if a

crash occurs. If you suspect a crash to be related to a dynamic problem such as heavy CPU load, voice-call or data traffic, please copy the performance data to your TFTP server immediately after the device came up again. This is because the collected information is stored at the maximum resolution only for the last hour. Next to this explicit way of exporting performance measures the supervisor log also stores the measures of the last 15 minutes before a crash.

The following command uploads the performance measures of the past month since the last power-cycle to a TFTP server.

Mode: Administrator execution

Step	Command	Purpose
1	<code>device#copy log:performance.tar tftp://<ip-addr>/<path>/performance.tar</code>	Uploads recent performance measures to a TFTP server.

Note Send the performance.tar file to our support team for further analysis.

Chapter 11 Programmable System-Event Configuration

Chapter contents

Introduction	151
System variables	151
User-Defined expressions	152
Actions	153
Expression configuration task list.....	153
Collect information about the variables that build the expression	154
Display existing system variables and expressions	154
Tracking real-time changes of system variables and expressions	155
Validate an expression (on-the-fly computation).....	155
Create/Modify an expression	156
Extend a system variable by an expression.....	157
Create/Modify an expression family.....	158
Wildcards and regular expressions in context-family names	159
Delete an expression	161
Expression Syntax.....	162
Data Types	162
Booleans	162
Numbers	162
Time Stamps	163
Text Strings	163
Errors	163
Operators	164
Logical Operators	164
Bitwise Operators	165
Arithmetic Operators	166
Comparison Operators	166
Operator Precedence	167
Functions	168
Logical Functions	169
Bitwise Functions	170
Arithmetic Functions	170
Comparison Functions	172
Set Functions	173
Time/Date Functions	173
Temporal Functions	174
Example Expressions	177
State Profiles.....	177
Default OVERLOAD state profile	179

Check the configuration of state profiles180
Debug state transitions181

Introduction

This chapter describes how to use programmable system events in Trinity. System events are triggered when a subsystem changes its state, for example if the link of an interface comes up. Trinity's event system is programmable: the administrator is able to combine existing events into new types of events. For example, it is possible to trigger the provisioning of a new software image when a custom event pattern occurs. Let us assume we want to provision a software image when the system is up and as soon as the management IP interface is up for some minutes and NTP is synchronized. This chapter explains how to construct and program such composite events.

The event programming system of Trinity is organized in three layers (System Variables, Expressions, and Actions) as depicted in [figure 16](#).

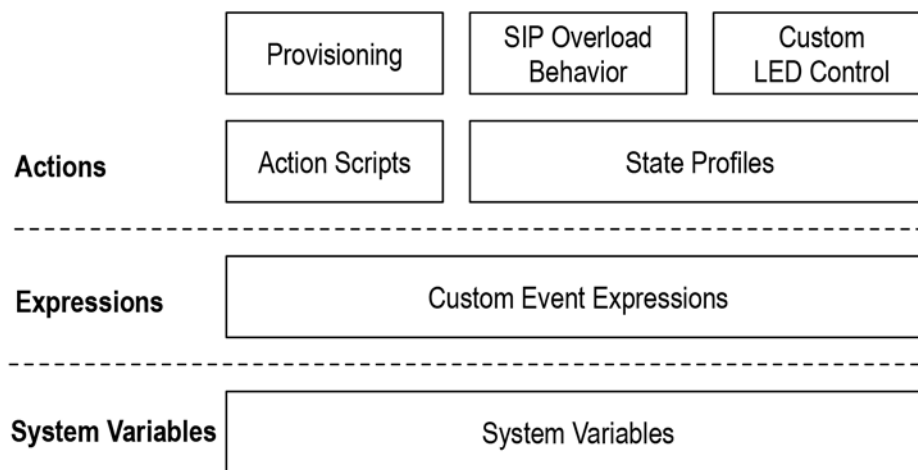


Figure 16. The Event-Programming System is organized in three layers

The remainder of this introduction gives a brief overview of these layers starting at the bottom. The sections following the introduction provide more detailed information and contain examples of how to configure the programmable event system.

System variables

Trinity exposes many subsystem states as system variables upon which action components can react. A system variable essentially is a name-value pair. The name uniquely identifies the variable and the value stores its current state. For example the system variable *sys.up* is initialized to *false*, but is changed to *true* as soon as the system is fully functional after boot-up.

[Table 2](#) lists the most important system variables that Trinity exposes to the user.

Table 2. Essential System Variables

Variable Name	Description
<code>ip.ctx:ctx-name.</code>	True if the IP interface <i>if-name</i> in IP context <i>ctx-name</i> is up; false if the IP interface is down.
<code>if:if-name.up</code>	

Table 2. Essential System Variables (Continued)

Variable Name	Description
ip.ctx:ctx-name. if:if-name. addr:addr-name.up	True if the IP address <i>addr-name</i> in IP interface <i>if-name</i> in IP context <i>ctx-name</i> is up; false if the IP address is down.
ntp.init	True if the time has been set by NTP.
ntp.sync	True if the time has been fully synchronized over NTP.
sys.cpu.load1m	CPU load, averaged over one minute, in percent.
sys.cpu.util1m	CPU utilization, averaged over one minute, in percent.
sys.ram.avail	Random-Access-Memory currently available in MB.
sys.up	True if the system is up and the startup-config has been applied.

As the name suggests system variables are generated by the system; they cannot be created or modified by the user.

Whenever the value of a variable changes a system event is triggered to all upper layers. Components that refer to the variable in order to execute an action (e.g. start provisioning, switch LEDs, etc.) are triggered automatically if the underlying variable changes.

User-Defined expressions

User-Defined expressions are composite variables created by the device administrator. An expression combines system variables and other expression using Trinity's powerful temporal expression algebra.

For example the administrator might want to know whether all of the following conditions are true (according to the provisioning example introduced above) when

- the system is up,
- the management IP interface is up for at least two minutes, and
- the device has synchronized its clock over NTP.

This overall condition is captured by the following expression entered as CLI configuration command:

```
expression READY "sys.up
                && DEBOUNCEINC(ip.ctx:ROUTER.if:WAN.up, 2m)
                && ntp.sync"
```

As depicted in [figure 17](#) on page 153 the expression creates a new variable called *READY*. Such composite variables are useful to build more complex expressions out of system variables. For example the administrator could create yet another expression called *NOT-READY*, which is true if the above conditions are not met:

```
expression NOT-READY "!READY"
```

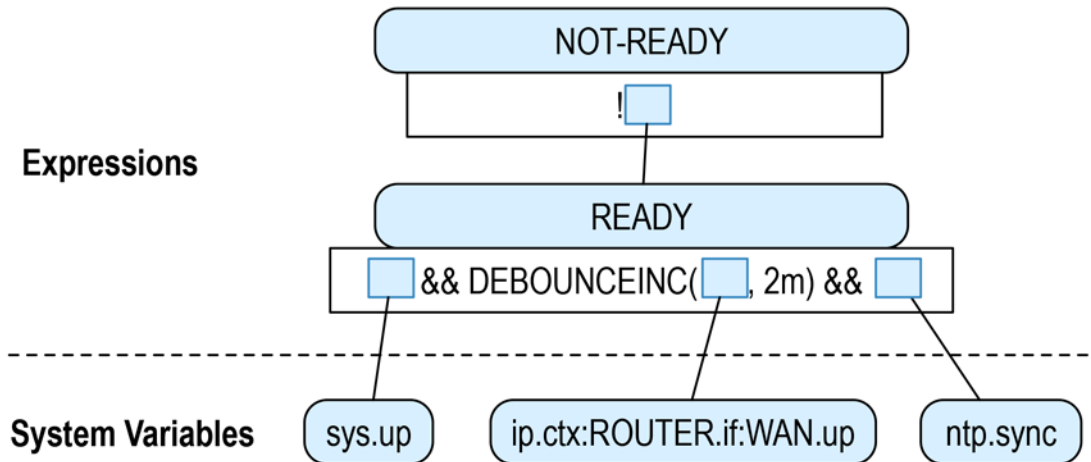


Figure 17. Expression Tree

Actions

Several Trinity components use system variables and expressions to control their behavior. For example the administrator may write an action script to execute a set of CLI commands whenever a variable changes its value from false to true. This powerful feature allows re-configuring virtually every aspect of the Patton device based on complex event patterns.

The following list briefly describes the components that can be linked to system variables and expressions. This list will be expanded in future releases of Trinity:

- **Action Scripts: NOT SUPPORTED YET.** Currently, action scripts are triggered by events that are not based on the programmable system events. Support will be added in one of the next releases.
- **SNMP Traps: NO SUPPORTED YET.** In the future, it will be possible to generate SNMP traps when a variable changes its value.
- **State Profiles:** Build simple state machines where variable changes trigger transitions between states. (See also section “[State Profiles](#)” on page 177.) State profiles map different events into well-defined and customizable states. For example, the *OVERLOAD* state-profile defines three states, *NORMAL*, *WARNING*, and *CRITICAL*, which reflect how the system resources are used based on the current CPU and memory utilization. The state profile is used by other components, for example the SIP call-limiter that drops incoming SIP messages if the system is overloaded.
- **SIP Overload Behavior:** You are able to configure in which circumstances the SIP user agent shall drop incoming and outgoing calls due to an overload situation. This feature uses a state profiles (see above) in combination with a description about which SIP messages to drop in which state. (For more information, refer to Chapter 51, “[SIP Overload Configuration](#)” on page 638)

Expression configuration task list

To configure user-defined expressions, perform the tasks in the following sections:

- “[Collect information about the variables that build the expression](#)” on page 154
- “[Validate an expression \(on-the-fly computation\)](#)” on page 155)

- “Create/Modify an expression” on page 156
- “Extend a system variable by an expression” on page 157
- “Create/Modify an expression family” on page 158
- “Delete an expression” on page 161

Collect information about the variables that build the expression

User-defined expressions are entered as mathematical equations that combine existing system variables and expressions. In order to build a new expression you need to know which variables to use.

Display existing system variables and expressions

The following CLI command lists all variables alphabetically by name. The second column of the table shows the current value of the variables.

Mode: Operator exec

Step	Command	Purpose
1	<code>node>show variable</code>	Displays a list of all system variables and user-defined expressions (names and values).
2	<code>node>show variable name</code>	Displays detailed information about the specified variable.

Example: Show all system variables and expressions and their current values:

```
Node>show variable

Variables
=====

Variable                                     Value
-----
ip.ctx:ROUTER.if:LAN.addr:LAN.up           true
ip.ctx:ROUTER.if:LAN.up                     true
ntp.init                                     true
ntp.sync                                     false
sip.pktq.delay                              0
sys.bootNr                                  440
sys.cpu.loadlm                              0
sys.cpu.util                                0
sys.cpu.util1m                              0
sys.initLevel                               100
sys.ram.avail                               158
sys.up                                       true
```

Example: Display all information available about a specific variable (sys.up):

```
node>show variable sys.up

Variable: sys.up
=====

Value:      true
```

Type: Boolean
 Data-Type: Boolean
 Description: True if the system is up and the config has been applied

Each variable has a data type and description assigned to it. The data type defines the range of values a variable can take. Next to the Boolean data type (true/false) there are data types for integer and floating-point numbers, text strings, and types for storing absolute time stamps and relative time differences. (For more information, refer to section “Data Types” on page 162.) Variables generated by the system also have a description assigned that describes the purpose and behavior of the variable.

Tracking real-time changes of system variables and expressions

The show commands presented above prints the current state of a variable. Trinity also offers a debug command with which you can observe value changes in real time:

Mode: Operator exec

Step	Command	Purpose
1	<code>node>debug variable [detail level full-detail]</code>	Enables the real-time debugger that prints information about changing variable values to the current terminal. Use detail-level 1 for a brief one-line statement for each variable change.

Example: Show all system variables and expressions and their current values:

```
node>debug variable detail 1
13:59:40.000 VAR      # [sys.cpu.util] 0 -> 2
13:59:50.000 VAR      # [sys.cpu.util] 2 -> 0
14:00:00.000 VAR      # [sys.cpu.util] 0 -> 25
14:00:00.000 VAR      # [sys.cpu.util] 1 -> 5
14:00:00.000 VAR      # [sys.ram.avail] 154 -> 150
14:00:10.000 VAR      # [sys.cpu.util] 25 -> 7
14:00:10.000 VAR      # [sys.cpu.util] 5 -> 6
```

The trace output above shows how the CPU load and available memory changes over a short period of time. Other than those performance values no variables were changed in the trace period.

Both the show and the debug commands are helpful instruments to debug your user-defined expressions.

Validate an expression (on-the-fly computation)

Before you add an expression to the system configuration permanently it is often useful to check whether the expression is syntactically valid. The CLI compute command can be used to compute an expression on the fly and print the result to the current terminal.

Mode: Operator exec

Step	Command	Purpose
1	<code>node>compute expression</code>	Computes the mathematical expression on-the-fly and displays the result.

Example: You may use `compute` instead of the `show variable` to quickly display the current value of a variable:

```
node>show variable sys.up
Variable: sys.up
=====

Value:      true
Type:       Boolean
Data-Type:  Boolean
Description: True if the system is up and the config
             has been applied

node>compute sys.up
true
```

Example: Furthermore, `compute` allows you to combine variables with mathematical expression. The following example computes some expressions, some of which are syntactically invalid or use variables that do not exist:

```
node>compute 1+2
3
node>compute 1/0
#DIV/0!
node>compute 1:2
% EXPRESSION ERROR:
  1:2
   ^ Unexpected character ':'
node>compute sys.up
true
node>compute !sys.up
false
node>compute sys.doesnotexist
#N/A!
```

Create/Modify an expression

Before you enter the expression command make sure you understand the meaning of the system variables you want to use. For example, if you want to use the `sys.ram.avail` variable recognize that the value represents the available memory in megabytes. Use the “**show variable** *name*” command to get information about a variable.

The procedure described below creates a new expression:

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg)#expression</code>	Creates a new variable called <i>name</i> the value of which is defined by the mathematical <i>expression</i> . If an expression variable with the same name already exists it is updated with the new mathematical <i>expression</i> specified.

An expression constitutes of two parts: a variable name and a mathematical expression that combines existing variables. Trinity exposes the expression as a variable that can be used in other expression. Therefore each

expression must be tagged with a unique name. You cannot re-use names of existing system variables or expressions.

The mathematical equation roughly follows Microsoft Excel formulas. (Refer to section “Operators” on page 164 for a list of all available operations and functions.) If you are familiar with Microsoft Excel you know how to compute the value of a cell based on the value of other cells by specifying the referred-to cells (e.g. A1+A2). Unlike Excel Trinity variables are not organized in a two-dimensional grid. In Trinity variables are identified by their name (e.g. MYVAR1+MYVAR2).

User-defined expression can also be used to define re-usable constants. The following example shows how to compute the area of a circle based on the constant PI and the radius R, both specified as expression variables:

Example: Define a constant PI, a variable *R*, and an expression *AREA* to compute the area of a circle. Change the variable *R* and show that the expression computing the area is updated accordingly.

```
node>enable
node#configure
node(cfg)#expression PI 3.14
node(cfg)#expression R 2
node(cfg)#expression AREA "PI*R*R"
node(cfg)#show variable
System Variables
=====
System Variable                               Value
-----
AREA                                           12.56
PI                                             3.14
R                                              2

node(cfg)#expression R 3
node(cfg)#show variable
System Variables
=====
System Variable                               Value
-----
AREA                                           28.26
PI                                             3.14
R                                              2
```

Extend a system variable by an expression

The name of Trinity system variables is structured in an object-oriented manner (e.g. *sys.up*, *ip.ctx:ROUTER.if:LAN.up*). A variable name consists of several nested objects separated by a dot (.). For example the *sys.up* variable can be regarded as variable up of object *sys*. If there are different instances of an object (e.g. different IP interfaces) the object is identified by its name after a colon (:). Thus the variable *ip.ctx:ROUTER.if:LAN.up* can be regarded as variable up of the object *if* with name *LAN* within the object *ctx* with name *ROUTER* within then object *ip*. This object structure often reflects the structure of the device configuration: “interface LAN” is configured within “context ip ROUTER”.

You are allowed to extend such an object by your own custom expressions. For example you may want to define an expression to delay the up-signal of the system.

Example: Add an expression to delay the up-signal of the system by one minute:

```
node(cfg)#expression sys.delayed-up DELAY(sys.up,1m)
```

This creates a new variable *delayed-up* in the object *sys*. If both the expression you create and the variables it refers to operate on the same object (e.g. *sys*) you may want to use the following shortcut notation:

```
node(cfg)#for sys expression delayed-up DELAY(up,1m)
node(cfg)#show variable
System Variables
=====
System Variable                               Value
-----
sys.delayed-up                                false
sys.up                                         true

a;sdlfk;laskdf;laskdf
```

This adds the “delayed-up” variable to the “sys” object and looks for referred-to variable (“up”) in the same “sys” object as well. This is summarized by the following procedure.

Mode: Configure

Step	Command	Purpose
1	<i>node(cfg)#for context expression name expression</i>	Creates a new variable called <i>context.name</i> . The value of the expression is defined by the mathematical <i>expression</i> . All variables mentioned in the expression are looked up in the context of the context object. If an expression variable with the same name already exists it is updated with the new mathematical <i>expression</i> specified.

Create/Modify an expression family

An expression family is a processing rule that is applied to a set of similar variables.

Example: Let us imagine you want to create an expression that delays the link state of every IP interface by 10 seconds. Instead of writing an expression for each individual IP interface

```
node(cfg)#expression ip.ctx:ROUTER.if:LAN.delayed-up \
    DELAY(ip.ctx:ROUTER.if:LAN.up)

node(cfg)#expression ip.ctx:ROUTER.if:WAN.delayed-up \
    DELAY(ip.ctx:ROUTER.if:WAN.up)

node(cfg)#expression ip.ctx:ROUTER.if:DMZ.delayed-up \
    DELAY(ip.ctx:ROUTER.if:DMZ.up)
```

Trinity allows you to define an expression family for all IP interfaces using the for expression syntax with wild-cards:

```
node(cfg)#for ip.ctx:%.if:% expression delayed-up
    DELAY(up)
```

The wildcard (%) matches all object names. Thus the string “if:%” matches to all our interfaces “if:LAN”, “if:WAN”, and “if:DMZ”. The command above creates a new variable “delayed-up” in each IP interface object and sets it to the delayed “up” variable of the IP interface as a value.

Let's repeat the syntax for the **for expression** command again with the emphasis that the *context* argument may be an object name containing wildcards or regular expressions.

Mode: Configure

Step	Command	Purpose
1	<i>node(cfg)#for context-family expression name expression</i>	Creates a family of new variable called <i>name</i> in any object that matches the <i>context-family</i> specification. The value of the expression is defined by the mathematical <i>expression</i> . All variables mentioned in the expression are looked up in the context of the corresponding context object. If an expression variable with the same name already exists it is updated with the new mathematical <i>expression</i> specified.

Note If the system creates a system variable that matches the *context-family* argument of an existing expression family, that expression family is automatically extended with an expression variable for the new system variable. For example let us assume you configured the expression family *delayed-up* above. If you create a new IP interface called *LAN2*, a new variable called *ip.ctx:ROUTER.if:LAN2.delayed-up* will be created automatically for the new interface.

Wildcards and regular expressions in context-family names

An expression is added for a whole family of variables if the *context-family* argument of the **for expression** command contains one or more wildcards or regular expressions:

- **Wildcard symbol (%)**: The wildcard symbol matches any character up to the next object separator (.) or object-name separator (:). For example the *context-family* string *a.%c* matches the object names
 - *a.b.c* (% ~= b),
 - *a.x.c* (% ~= x),
 - *a.test.c* (% ~= test),
 - but not match *a.b.b.c* (% != b.b)
- **Regular expression ({}):** If you need more complex matching rules you can write a regular expression in curly braces. For example the *context-family* string *a.{test\d+}.c* matches the object names
 - *a.test1.c* (test\d+ ~= test1),
 - *a.test2.c* (test\d+ ~= test),
 - *a.test.c* (test\d+ != test),

The regular expression itself may use the following tokens:

Table 3. Regular Expression Tokens

Symbol	Description	Example
.	Matches any single character.	
[<i>charset</i>]	Matches a single character that is contained within the brackets. [abc] matches “a”, “b”, or “c”. [a-z] specifies a range which matches any lowercase letter from “a” to “z”. These forms can be mixed: [abc1-9] matches “a”, “b”, “c”, or any digit. The “-” character is treated as a literal character if it is the last or the first. For example, [abc-] matches “a”, “b”, “c”, or “-”.	LAN[0-9] matches any string starting with the prefix “LAN” followed by a digit.
[^ <i>charset</i>]	Matches a single character that is not contained within the brackets. [^abc] matches any character other than “a”, “b”, or “c”. [^a-z] matches any single character that is not a lowercase letter from “a” to “z”.	[^+]+ matches any non-empty string without the dash symbol
()	Groups a subexpression.	(test)+ matches any repetition of the string “test”, e.g. “testtesttest”.
\d	Matches any digit, same as [0-9]	LAN\d+ matches {“LAN1”, “LAN2”, etc.}
\D	Matches any character other than a digit, same as [^0-9].	
\u	Matches any uppercase letter, same as [A-Z].	
\U	Matches any character other than an uppercase letter, same as [^A-Z].	
\l	Matches any lowercase letter, same as [A-Z].	
\L	Matches any character other than a lowercase letter, same as [^A-Z].	
\w	Matches any letter, same as [A-Za-z].	
\W	Matches any character other than a letter, same as [^A-Za-z].	
[<i>class</i> :]	Matches any character from the specified character class. The following character classes exist and are representing the following character group: <ul style="list-style-type: none"> • alnum[0-9A-Za-z] • alpha[A-Za-z] • blank[] • digit[0-9] • lower[a-z] • puncte.g. “.”, “,”, “@”, etc. • upper[A-z] • word[0-9A-Za-z] • xdigit[0-9A-Fa-f] 	
?	Matches zero or one occurrences of the preceding element.	test? ~={“tes”, “test”}

Table 3. Regular Expression Tokens (Continued)

Symbol	Description	Example
%	Matches zero or more occurrences of the preceding element.	test% =~ {"tes", "test", "testttttt"}
+	Matches one or more occurrences of the preceding element.	test+ =~ {"test", "testt", "testtttt"}

Example: Create an expression that returns true only if all LAN IP interfaces are up in the default context ROUTER. We assume that all LAN IP interface follow the naming format LAN1, LAN2, etc.

```
node(cfg)#for ip.ctx:ROUTER expression all-if-up \
                                AND(if:{LAN\d+}.up)
```

Note The curly brackets `{}` are used to tell the CLI that whatever is between the brackets is a regular expression. The curly brackets themselves are not part of the regular expression. In the example above, the regular expression is `LAN\d+`.

Delete an expression

Use one of the following commands to:

- Delete an expression by its name
- Delete an expression within an object context
- Delete an entire expression family

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg)#no expression name</code>	Deletes the expression called <i>name</i> .
2	<code>node(cfg)#no for context expression name</code>	Deletes the expression <i>context.name</i> .
3	<code>node(cfg)#no for context-family expression name</code>	Deletes the expression <i>name</i> for the set of all contexts specified by the <i>context-family</i> .

Note Trinity does not prevent you from deleting an expression variable that is referred to by another expression. If an expression uses a non-existing variable, its value will be re-computed to an `#N/A!` error.

Example: Delete the constant PI that is used by the expression AREA.

```
node(cfg)#no expression PI
node(cfg)#show variable
System Variables
=====
System Variable                               Value
-----
AREA                                           #N/A!
R
```

Expression Syntax

Trinity expressions are used to combine system variables with a mathematical expression. An expression may contain the following parts:

- Constants (The Boolean values true or false, numbers, time stamps, text strings)
- References (Names of other variables as placeholders for their values)
- Operators (The '+' operator sums two values)
- Functions (The NOW() function returns the current date and time)

Data Types

Constants are parts of the expression that are not calculated. Consider the expression "3.14*R*R" where "3.14" is a floating-point constant. Trinity expressions are type-aware. This means that each constant is assigned to a certain data type. We will discuss those data types and some of its typical constant values below.

Booleans

The Boolean data type only has two values (**true** and **false**). Boolean values represent the truth-values of logic operations (e.g. "true && false" => false). You enter Boolean constants by using one of the following words:

- false
- true

Numbers

Number data types come in two flavors, integers (whole numbers) and floating-point numbers (e.g. 3.14). Trinity automatically takes care of using the adequate flavor for each operation, so you don't have to convert manually between integers and floating-point numbers. You can enter numeric constants with different notations as shown below:

- Integer constants:
 - 123 (decimal notation)
 - 0xff(=255; hexadecimal notation; "0x" prefix)
 - 010 (=8; octal notation; "0" prefix)
- Floating-point constants:
 - 3.14 (floating-point constant)
 - 2.3e-3 (=0.0023; exponent notation)
- Integer and floating-point constants can be used in any order when building expressions:
 - 3.14+2 => 5.14
 - 1.1+2.2 => 3.3
 - 1.1-0.1 => 1

Time Stamps

Trinity distinguishes between absolute time stamps (date/time) and relative time stamps (time difference). Below you find examples how to enter time stamp constants:

- Clock (absolute time stamps): Entered in ISO time notation (yyyy-mm-ddThh:mm:ss)
 - 2016-01-11T12:40:00 (date and time in time zone of the local clock)
 - 2016-01-11 (date only)
 - 12:40:00 (time of day only)
- Time (relative time difference): Entered as a combination of days (d), hours (h), minutes (m), seconds (s), milliseconds (ms), microseconds (us), and nanoseconds (ns).
 - 1d2h (one day plus two hours)
 - 2m30s (two and a half minutes)
 - 1s500ms (one and a half second)
 - In expressions relative time constants can be added to or subtracted from absolute time stamps. The result is a new absolute time stamp:
 - 2016-01-11+1d => 2016-01-12
 - 12:40:00+2m => 12:40:02

Text Strings

Variables can also be used to store text strings. Thus it is important to be able to enter text constants in expression, for example to compare the strings. Text strings must be put in single quotes (') to distinguish them from variable names.

- 'This is a test'

Strings can be concatenated by the (+) operator and repeated by the (*) operator

- 'abc'+ 'def' => 'abcdef'
- 'x'*4 => 'xxxx'

Errors

Errors cannot be entered as constants but may results as computation value. [Table 4](#) shows all possible computation errors and their meanings:

Table 4. Error Values

Error Value	Description
#DIV/0!	Division by zero: Result of an expression that contains a division by zero operation, e.g. "2/(1-1)".
#N/A!	Invalid reference: Result of an expression that refers to a variable that does not exist. Also if your expression uses the result of another expression and you delete that second expression, the first expression will result in this error.
#NUM!	Invalid number: An operator or function is expecting a number but finds another data type, e.g. "2+'test'".

Table 4. Error Values (Continued)

Error Value	Description
#VAL!	Invalid value: A function for example expects a positive number but finds a negative number as argument.

Operators

Operators are symbols in your equation that specify the calculation you want to perform on the left-hand side and the right-hand side of the operator. The following tables list all available operators available to Trinity expressions grouped by type.

Logical Operators

Logical operators are used to reason about logical statements.

Table 5. Logical Operators

Operator	Description	Example
!	Unary negation operator. Returns true if the value that follows the operator evaluates to false, returns false if the value evaluates to true.	!true => false
 or	Logical OR operator. Returns true if either the left-hand side value or the right-hand side value evaluates to true. You can either use the symbolic notation or the keyword "or".	true false => true true or false => true
^^ xor	Logical XOR operator. Returns true if either the left-hand side value or the right-hand side value but not both evaluate to true. You can either use the symbolic notation or the keyword "xor".	true ^^ true => false true xor true => false
&& and	Logical AND operator. Returns true if both the left-hand side value and the right-hand side value evaluate to true. You can either use the symbolic notation or the keyword "and".	true && false => false true and false => false

If the right-hand or the left-hand side value of a logical operator is not of a Boolean data type the value is automatically converted to true or false according to the following rules:

- **Numbers:** Non-zero numbers are evaluated to true, zero is evaluated to false.
- **Absolute Timestamps:** Only the epoch (1970-01-01:00:00:00) is evaluated to false; all other timestamps are evaluated to true.
- **Relative Timestamps:** Zero time delta values (e.g. 0s) is evaluated to false, non-zero time delta values (e.g. 500ms) are evaluated to true.
- **Text strings:** An empty string ('') is evaluated to false whereas all non-empty strings (e.g. 'test') are evaluated to true.

Examples:

- `1 && 2 => true`
- `0 || 'test' => true`
- `1m && 0s => false`

Bitwise Operators

Bitwise operators operate on number data types only and apply a Boolean operation bit by bit.

Table 6. Bitwise Operators

Operator	Description	Example
<code>~</code>	Unary bitwise complement operator. Returns the bit-wise complement of a number, i.e. turns zero-valued bits into one-valued bits and vice-versa	<code>~0xffffffff => 0</code>
<code> </code>	Bitwise OR operator. Sets each result bit to one if one of the corresponding bits in the left- and right-hand side values is one; sets the result bit to zero otherwise.	<code>0x11 0x01 => 0x11</code>
<code>^</code>	Bitwise XOR operator. Sets each result bit to one if one but not both of the corresponding bits in the left- and right-hand side values are one; sets the result bit to zero otherwise.	<code>0x11 ^ 0x01 => 0x10</code>
<code>&</code>	Bitwise AND operator. Sets each result bit to one if both corresponding bits in the left- and right-hand side values are one; sets the result bit to zero otherwise.	<code>0x11 & 0x01 => 0x01</code>
<code><<</code>	Bitwise left shift: Shifts the number on the right-hand side of the operator left by the number of bits specified with the right-hand side value.	<code>0x01 << 4 => 0x10</code>
<code>>></code>	Bitwise right shift: Shifts the number on the right-hand side of the operator right by the number of bits specified with the right-hand side value.	<code>0x10 >> 4 => 0x01</code>

Arithmetic Operators

Table 7. Arithmetic Operators

Operator	Description	Example
+	<p>Sum. Computes the sum of the left-hand side and the right-hand side value.</p> <p>If both values are strings, the sum operator concatenates the strings.</p> <p>If one of the values is an absolute time-stamp and the other is a relative time delta, the result is an absolute timestamp.</p>	<p>1+2 => 3</p> <p>'abc'+ 'def' => 'abcdef'</p> <p>2015-01-10+2d => 2015-01-12</p>
-	<p>Difference. Computes the difference between the left-hand side and the right-hand side value.</p> <p>If the left-hand side value is an absolute timestamp and the right-hand side value is a relative time delta, the result is an absolute timestamp.</p>	<p>1-2 => -1</p> <p>2015-01-10-2d => 2015-01-08</p>
*	<p>Product. Computes the product of the left-hand side and the right-hand side value.</p> <p>If the left-hand side value is a text string and the right-hand side value is an integer number, the result is the string concatenated n times to itself.</p> <p>If one argument is a relative time delta and the other argument is an integer number, the result is the time delta multiplied n times.</p>	<p>2*3 => 6</p> <p>'x'*3 => 'xxx'</p> <p>30m*2 => 1h</p>
/	<p>Division. Computes the division between the left-hand side and the right-hand side value.</p> <p>If the left-hand side value is a relative time delta and the right-hand side is an integer, the result is a relative time delta.</p>	<p>5/2 => 2.5</p> <p>1d/2 => 12h</p>
%	<p>Modulo. Computes the remainder from the integer division of the left-hand side value and the right-hand side value.</p>	<p>13%5 => 3</p>

Comparison Operators

Table 8. Comparison Operators

Operator	Description	Example
==	<p>Equal. Returns true if the left-hand side value is equal to the right-hand side value.</p> <p>Note: Data types are not converted automatically.</p>	<p>1 == 2 => false</p> <p>(1+1) == 2 => true</p> <p>1 == '1' => false</p>

Table 8. Comparison Operators (Continued)

Operator	Description	Example
!=	Not equal. Returns true if the left-hand side value is not equal to the right-hand side value. Note: Data types are not converted automatically.	1 != 2 => true (1+1) != 2 => false 1 != '1' => false
<	Less than. Returns true if the left-hand side value is less than the right-hand side value. If both arguments are strings, the operator performs a lexical comparison. If the arguments are not of the same data type an #VAL! error is returned.	1 < 2 => true 1 < 1 => false 'abc' < 'def' => true 1 < '1' => #VAL!
>	Greater than. Returns true if the left-hand side value is greater than the right-hand side value. If both arguments are strings, the operator performs a lexical comparison. If the arguments are not of the same data type an #VAL! error is returned.	1 > 2 => false 1 > 1 => false 'abc' > 'def' => false 1 > '1' => #VAL!
<=	Less than or equal. Returns true if the left-hand side value is less than or equal the right-hand side value. If both arguments are strings, the operator performs a lexical comparison. If the arguments are not of the same data type an #VAL! error is returned.	1 <= 2 => true 1 <= 1 => true 'abc' <= 'def' => true 1 <= '1' => #VAL!
>=	Greater than or equal. Returns true if the left-hand side value is greater or equal than the right-hand side value. If both arguments are strings, the operator performs a lexical comparison. If the arguments are not of the same data type an #VAL! error is returned.	1 >= 2 => false 1 >= 1 => true 'abc' >= 'def' => false 1 > '1' => #VAL!

Operator Precedence

The order in which Trinity evaluates operators is well defined according to the table below. For example, the products operator (*) is evaluated before the sum operator (+):

$$2+2*2 \Rightarrow 6$$

You can change this order by using parentheses:

$$(2+2)*2 \Rightarrow 8$$

If the equation contains operators with the same precedence they are evaluated from left to right:

$$2+2-2 \Rightarrow (2+2)-2 \Rightarrow 3$$

The following table shows the order in which operators are evaluated. The operators in the top row are evaluated first, those in the bottom row at the end.

Table 9. Operator Precedence

Operator	Description
! not ~ -	Unary operators (Boolean negation, bitwise complement, arithmetic negation)
* / %	Multiplication, division, and modulo operators
<< >>	Bit shift operators
< > <= >=	Comparison operators except equal and not equal
== !=	Equal and not-equal comparison operators
&	Bitwise AND operator
^	Bitwise XOR operator
	Bitwise OR operator
&& and	Logical AND operator
^^ xor	Logical XOR operator
 or	Logical OR operator

Functions

Functions are predefined equations that perform calculations on their arguments. Functions start with the upper-case function name, followed by an open parenthesis, an list of comma-separated arguments, and a closing parenthesis, e.g.

- DAY(2015-01-12) => 12
- BOOL(123) => true
- POWER(2,10) => 1024
- SQRT(9) => 3
- COUNT(1,2,3,4,5) => 5

Functions can be nested. That is instead of specifying a argument with a constant or variable name you can use another function:

- DAY(NOW()) => 11
- BOOL(SQRT(9)) => true
- COUNT(NOW(),SQRT(9)) => 2

The following tables lists all available functions available to Trinity expressions grouped by type.

Logical Functions

All logical operators (not, or, xor, and) are also available as functions. So instead of writing the expression "a && b" you may write the expression "AND(a,b)". The advantage of the function is that it may take more than two arguments, e.g. "AND(a,b,c,d)", which evaluates to true only if all arguments are true.

All logical functions take one or more arguments (except NOT(), which takes exactly one), convert all arguments to a Boolean value and returns the logical operation on all the arguments. If no argument is provided an invalid-value error (#VAL!) is returned.

- AND(1,2,3) => true
- AND(1,2,3,0) => false
- AND() => #VAL!

If an argument specifies a set of variables by using a regular expression, the regular expression is expanded and all matching variables are passed to the function. For example if there are three variables "test:1", "test:2", "test:3" the expression "AND(test:%)" is expanded to "AND(test:1,test:2,test:3)".

Table 10. Logical Functions

Function	Description	Example
NOT(boolean)	Negation. Returns true if the boolean argument evaluates to false, returns false if the argument evaluates to true.	NOT(false) => true NOT(true) => false
OR(boolean1[,boolean2...])	Logical OR operator. Returns true if at least one of the arguments evaluate to true.	OR(1,2,3) => true OR(1,2,0) => false OR() => #VAL
XOR(boolean1[,boolean2...])	Logical XOR operator. Returns true if an odd number of arguments evaluate to true.	XOR(1,2,3) => true XOR(1,2,0) => false XOR() => #VAL
AND(boolean1[,boolean2...])	Logical AND operator. Returns true if both all arguments evaluate to true.	AND(1,2,3) => true AND(1,2,0) => false AND() => #VAL
BOOL(arg)	Converts the argument to a Boolean value. Converts error to the value false. Use this function if you don't want errors in used variables to be passed as errors to the result of the expression.	BOOL(1/0) => false BOOL('test') => true BOOL('') => false

Bitwise Functions

The bitwise operators (`-`, `|`, `^`, `&`) are also available as functions. So instead of writing the expression "`a & b`" you may write the expression "`BITAND(a,b)`". The advantage of the function is that it may take more than two arguments, e.g. "`BITAND(a,b,c,d)`", sets a bit in the result to true only if the same bit is true in all arguments.

If an argument specifies a set of variables by using a regular expression, the regular expression is expanded and all matching variables are passed to the function. For example if there are three variables "`test:1`", "`test:2`", "`test:3`" the expression "`BITAND(test:%)`" is expanded to "`BITAND(test:1,test:2,test:3)`".

Table 11. Bitwise Functions

Function	Description	Example
BITNOT (<i>number</i>)	Bitwise complement. Returns the bit-wise complement of a number, i.e. turns zero-valued bits into one-valued bits and vice-versa	BITNOT(0xffffffff) => 0
BITOR (<i>number</i> [, <i>number...</i>])	Bitwise OR. Sets each result bit to one if one of the corresponding bits in any of the arguments is one; sets the result bit to zero otherwise.	BITOR(0x1,0x2) => 0x3 BITOR() => #VAL
BITXOR (<i>number</i> [, <i>number...</i>])	Bitwise XOR. Sets each result bit to one if the corresponding bits in an odd number of arguments are one; sets the result bit to zero otherwise.	BITXOR(0x1,0x2) => 0x3 BITXOR() => #VAL
BITAND (<i>number</i> [, <i>number...</i>])	Bitwise AND. Sets each result bit to one if the corresponding bits in all arguments is one; sets the result bit to zero otherwise.	BITAND(0x1,0x2) => 0x0 BITAND() => #VAL
BITLSHIFT (<i>number</i> , <i>shift</i>)	Bitwise left shift: Shifts the number argument by shift bits to the left.	BITLSHIFT(0x1,4) => 0x10
BITRSHIFT (<i>number</i> , <i>shift</i>)	Bitwise right shift: Shifts the number argument by shift bits to the right.	BITRSHIFT(0x10,4) => 0x1

Arithmetic Functions

Table 12. Arithmetic Functions

Function	Description	Example
ABS (<i>number</i>)	Returns the absolute value of a <i>number</i>	ABS(123) => 123 ABS(-3.14) => 3.14
CEILING (<i>number</i> [, <i>significance</i>])	Rounds a <i>number</i> up to the nearest integer or to the nearest multiple of <i>significance</i> . The default <i>significance</i> is 1.	CEILING(3.14,0.1) => 3.2
DIFF (<i>minuend</i> , <i>subtrahend</i>)	Subtracts the <i>subtrahend</i> from the <i>minuend</i> (same as the '-' operator). If the minuend is an absolute timestamp and the subtrahend is a relative time delta, the result is an absolute timestamp.	DIFF(1,2) => -1 DIFF(2015-01-10,2d) => 2015-01-08

Table 12. Arithmetic Functions (Continued)

Function	Description	Example
DIV (<i>dividend</i> , <i>divisor</i>)	Divides the <i>dividend</i> by the <i>divisor</i> (same as the '/' operator). If the <i>divisor</i> evaluates to zero the function returns a #DIV/0! error. If the <i>dividend</i> is a relative time delta and the divisor is an integer, the result is a relative time delta.	DIV(5,2) => 2.5 DIV(1/0) => #DIV/0! DIV(1d,2) => 12h
FLOOR (<i>number</i> [, <i>significance</i>])	Rounds a <i>number</i> down to the nearest integer or to the multiple of <i>significance</i> . The default <i>significance</i> is 1.	FLOOR(3.14,0.1) => 3.1
INT (<i>number</i>)	Rounds a <i>number</i> down to the nearest integer.	INT(3.14) => 3
MOD (<i>dividend</i> , <i>divisor</i>)	Computes the remainder from the integer division of the <i>dividend</i> and the <i>divisor</i> . If the <i>divisor</i> evaluates to zero the function returns a #DIV/0! error.	MOD(13,5) => 3
MROUND (<i>number</i> [, <i>multiple</i>])	Returns the <i>number</i> rounded to the desired <i>multiple</i> . The default <i>multiple</i> is 1.	MROUND(3.25,0.1) => 3.3
NEG (<i>number</i>)	Returns the negative <i>number</i> .	NEG(2+2) => -4
POWER (<i>number</i> , <i>power</i>)	Raises the <i>number</i> to the specified <i>power</i> .	POW(3,2) => 9
PRODUCT (<i>factor</i> [, <i>factor</i> ...])	Computes the product of all factors. If the first argument is a text string and the second a positive integer this function returns the text concatenated n times to itself.	PRODUCT(1,2,3,4,5) => 120 PRODUCT('x',3) => 'xxx' PRODUCT() => #VAL!
QUOTIENT (<i>dividend</i> , <i>divisor</i>)	Returns the integral part of the division of the <i>dividend</i> and the <i>divisor</i> . If the <i>divisor</i> evaluates to zero the function returns a #DIV/0! error.	QUOTIENT(9,2) => 4
MROUND (<i>number</i> [, <i>digits</i>])	Returns a <i>number</i> argument rounded to the desired fractional <i>digits</i> ; the default number of <i>digits</i> is 0	MROUND(3.25,1) => 3.3 MROUND(123,-2) => 100
SQRT (<i>number</i>)	Computes the square root of the <i>number</i> argument.	SQRT(16) => 4

Table 12. Arithmetic Functions (Continued)

Function	Description	Example
SUM (<i>summand</i> [, <i>summand</i> ...])	Computes the sum over all <i>number</i> arguments. If all arguments are text strings the function concatenates the strings.	SUM(1,2,3,4,5) =>15 SUM('a','b','c') => 'abc' SUM() => #VAL!

Comparison Functions

Table 13. Comparison Functions

Function	Description	Example
EQ (<i>arg1</i> , <i>arg2</i>)	Returns true if the two arguments are equal (equal type and equal value) (same as the '=' operator).	EQ(1,2) => false EQ(1d,60m) => true
NEQ (<i>arg1</i> , <i>arg2</i>)	Returns true if the two arguments are not equal (unequal type or unequal value) (same as the '!=' operator).	NEQ(1,2) => true NEQ(1d,60m) => false
IF (<i>condition</i> , <i>true-expr</i> , <i>false-expr</i>)	Returns the <i>true-expr</i> argument if the condition evaluates to true, else returns the <i>false-expr</i> parameter.	DIFF(1,2) => -1 DIFF(2015-01-10,2d) => 2015-01-08
ISERR (<i>arg</i>)	Returns true if the argument <i>arg</i> is an error.	ISERR(1/1) => false ISERR(1/0) => true
LE (<i>arg1</i> , <i>arg2</i>)	Less than or equal. Returns true if <i>arg1</i> is less than or equal <i>arg2</i> (same as the '<=' operator). If both arguments are strings, the function performs a lexical comparison. If the arguments are not of the same data type an #VAL! error is returned.	LE(1,2) => true LE(1,1) => true LE('abc','def') => true LE(1,'1') => #VAL!
LT (<i>arg1</i> , <i>arg2</i>)	Less than. Returns true if <i>arg1</i> is less than <i>arg2</i> (same as the '<' operator). If both arguments are strings, the function performs a lexical comparison. If the arguments are not of the same data type an #VAL! error is returned.	LT(1,2) => true LT(1,1) => false LT('abc','def') => true LT(1,'1') => #VAL!
GE (<i>arg1</i> , <i>arg2</i>)	Greater than or equal. Returns true if <i>arg1</i> is greater or equal than <i>arg2</i> (same as the '>=' operator). If both arguments are strings, the function performs a lexical comparison. If the arguments are not of the same data type an #VAL! error is returned.	GE(1,2) => false GE(1,1) => true GE('abc','def') => false GE(1,'1') => #VAL!

Table 13. Comparison Functions (Continued)

Function	Description	Example
GT(<i>arg1</i>, <i>arg2</i>)	Greater than. Returns true if <i>arg1</i> is greater than <i>arg2</i> (same as the '>' operator). If both arguments are strings, the function performs a lexical comparison. If the arguments are not of the same data type an #VAL! error is returned.	GT(1,2) => false GT(1,1) => false GT('abc','def') => false GT(1,'1') => #VAL!

Set Functions

These functions return information about the number of arguments but don't look at the arguments themselves.

Table 14. Set Functions

Function	Description	Example
COUNT([<i>arg*</i>])	Counts how many values are in the list of arguments	COUNT(1,false,4,1,0) => 5 COUNT(ip.ctx:ROUTER.if:%.up) => counts the number of IP interfaces in context ROUTER
COUNTIF([<i>arg*</i>])	Counts how many values are in the list of arguments that evaluate to true	COUNTIF(1,false,4,1,0) => 3 COUNTIF(ip.ctx:ROUTER.if:%.up) => counts the number of IP interfaces in context ROUTER that are up

Time/Date Functions

Table 15. Time/Date Functions

Function	Description	Example
DATE(<i>year</i>, <i>month</i>, <i>day</i>)	Builds a date value from the specified <i>year</i> , <i>month</i> , and <i>day</i> number arguments. This function is most useful in situations where the year, month, and day are supplied by formulas or references.	DATE(2015,12,11) => 2015-12-11
DATETIME(<i>date</i>, <i>time</i>)	Builds a date/time value from the specified <i>date</i> and <i>time</i> arguments. This function is most useful in situations where the date and time values are supplied by formulas or references.	DATETIME(TODAY(), 12:00:00) => 2016-01-11T12:00:00
DAY(<i>date-time</i>)	Returns the days part of a date/time value	DAY(2015-12-13T10:20:30) => 13
EPOCH()	Returns the earliest date/time value possible.	EPOCH() => 1970-01-01T00:00:00

Table 15. Time/Date Functions (Continued)

Function	Description	Example
HOUR (<i>date-time</i>)	Returns the hours part of a date/time value	DAY(2015-12-13T10:20:30) => 10
MINUTE (<i>date-time</i>)	Returns the minutes part of a date/time value	DAY(2015-12-13T10:20:30) => 20
MONTH (<i>date-time</i>)	Returns the months part of a date/time value	DAY(2015-12-13T10:20:30) => 12
NOW ()	Returns the current date/time in the local timezone; note that if NOW() is used in an expression, that expression is only updated every 10 seconds.	NOW() => 2016-01-11T19:05:00
SECOND (<i>date-time</i>)	Returns the seconds part of a date/time value	DAY(2015-12-13T10:20:30) => 30
TIME (<i>hour, minute, second</i>)	Builds a time value from the specified <i>hour</i> , <i>minute</i> , and <i>second</i> number arguments. This function is most useful in situations where the <i>hour</i> , <i>minute</i> , and <i>second</i> are supplied by formulas or references.	TIME(10,11,12) => 10:11:12
TODAY ()	Returns the current date in the local timezone; note that if TODAY() is used in an expression, that expression is only updated every 10 seconds.	TODAY() => 2016-01-11
YEAR (<i>date-time</i>)	Returns the year part of a date/time value	DAY(2015-12-13T10:20:30) => 2015

Temporal Functions

The following functions allow to reason about the temporal relation of events. Temporal functions also provide means of temporally manipulate event occurrences, for example delaying events.

Table 16. Temporal Functions

Function	Description
DEBOUNCE (<i>arg, period</i>)	De-bounces the argument <i>arg</i> by the specified time period, i.e. filters out frequent changes of <i>arg</i> . If <i>arg</i> changes, the change is not passed to the output immediately but is rather delayed by the specified period. If <i>arg</i> changes back to the original value within the period (unlike the DELAY function) the spurious value is never passed to the output. Figure 18 on page 176 shows a time diagram of an input argument <i>x</i> , de-bounced by 500ms.

Table 16. Temporal Functions

Function	Description
DEBOUNCEDEC (<i>arg, period</i>)	<p>Only de-bounces the argument <i>arg</i> by the specified time period if <i>arg</i> is decreased, i.e., if the new value is lower (e.g. true -> false). If <i>arg</i> is increased (e.g. false -> true) the value of <i>arg</i> is always passed to the output immediately. Figure 18 on page 176 shows a time diagram of an input argument <i>x</i> and its decrease de-bounced by 500ms.</p> <p>This function is useful to trigger a link-down event only after a certain time and only if the link does not come up again in this period:</p> <p>DEBOUNCEDEC(ip.ctx:ROUTER.if:LAN, 1m)</p>
DEBOUNCEINC (<i>arg, period</i>)	<p>Only de-bounces the argument <i>arg</i> by the specified time period if <i>arg</i> is increased, i.e., if the new value is higher (e.g. false -> true). If <i>arg</i> is decreased (e.g. true -> false) the value of <i>arg</i> is always passed to the output immediately. Figure 18 on page 176 shows a time diagram of an input argument <i>x</i> and its increase de-bounced by 500ms.</p> <p>This function is useful to trigger a link-up event only after a certain time and only if the link does not go down again in this period:</p> <p>DEBOUNCEINC(ip.ctx:ROUTER.if:LAN, 1m)</p>
DEBOUNCEINC (<i>arg, period, condition</i>)	<p>Only de-bounces the argument <i>arg</i> by the specified time period if the condition evaluates to true. While the condition evaluates to false the value of <i>arg</i> is always passed to the output immediately. Figure 18 on page 176 shows a time diagram of an input argument <i>x</i>, a condition <i>c</i>, and its output de-bounced by 500ms.</p>
DELAY (<i>arg, period</i>)	<p>Delays the argument <i>arg</i> by the specified time period. Figure 19 on page 176 shows a time diagram of an input argument <i>x</i> delayed by 500ms.</p>
ONCE (<i>arg</i>)	<p>Changes to true once when the argument <i>arg</i> evaluates to true, but never changes back to false until the device is rebooted. Figure 19 on page 176 shows a time diagram of an input argument <i>x</i> passed through the ONCE function.</p> <p>This function is useful to make sure a certain action is only executed once after boot-up, for example after the device synchronized its clock over NTP:</p> <p>ONCE(ntp.sync)</p>
STABLE (<i>arg, period</i>)	<p>Returns false as soon as <i>arg</i> changes. Switches to true only when the argument <i>arg</i> is stable for the specified <i>period</i> of time. Figure 19 on page 176 shows a time diagram of the STABLE function applied to an input argument <i>x</i> by 500ms.</p>

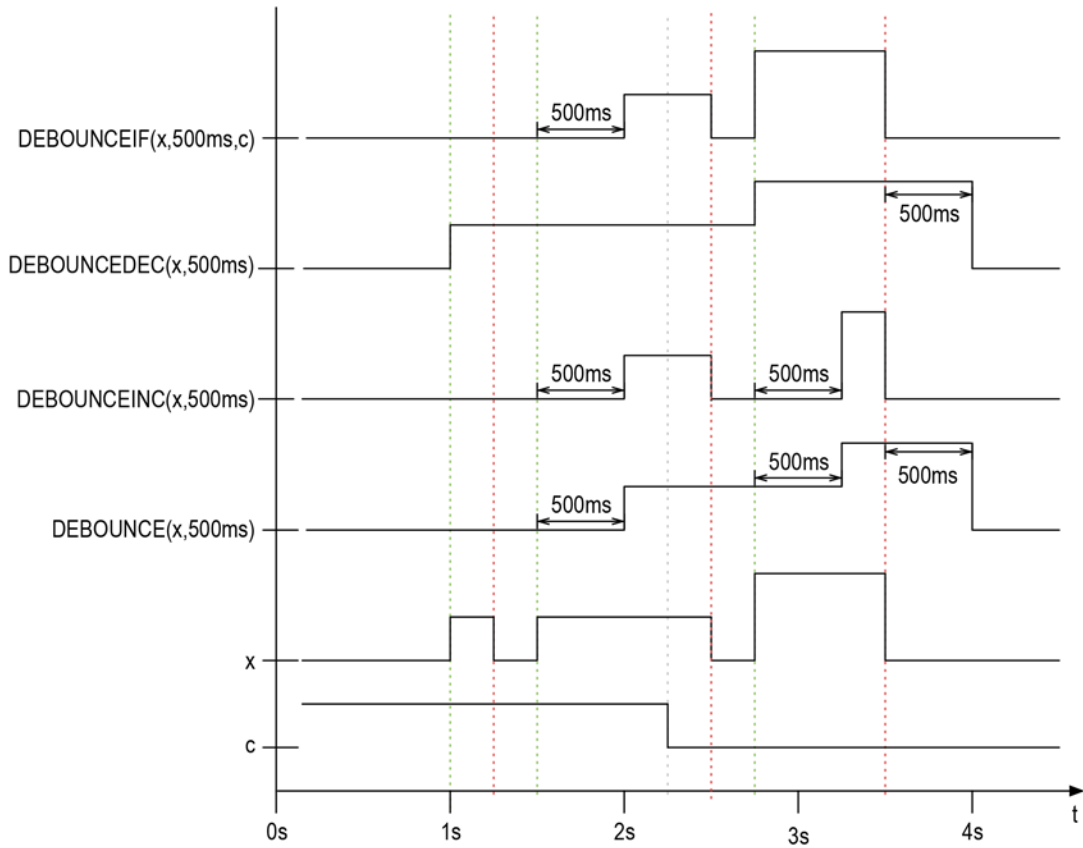


Figure 18. Time Diagram of the DEBOUNCE Functions

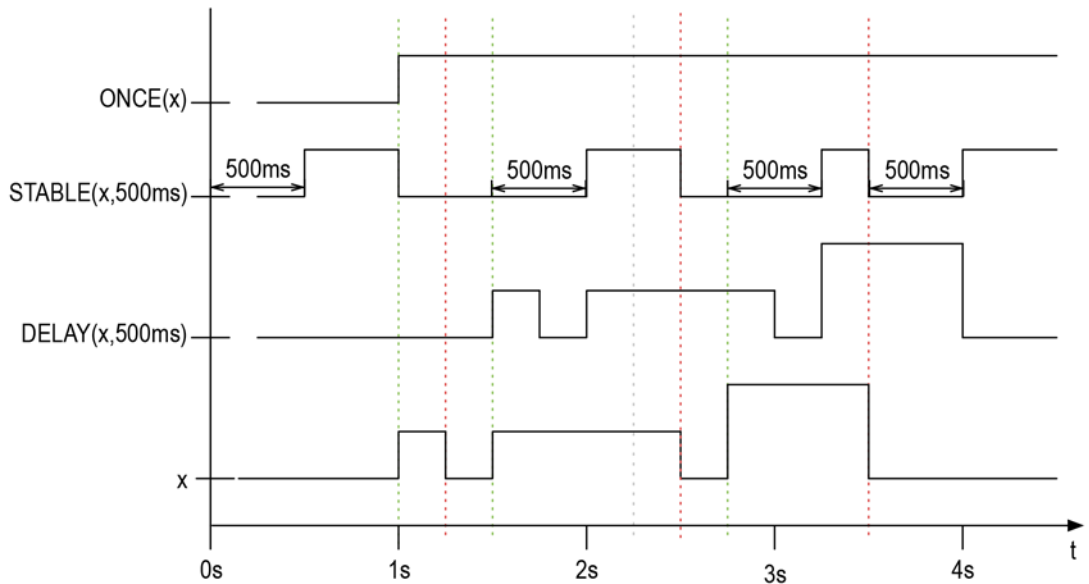


Figure 19. Time Diagram of the Functions ONCE, STABLE, and DELAY

Example Expressions

The following section lists some expression example that may serve as hints of how to use the expression algebra to build powerful temporal expressions.

Example: De-Bouncing link down-up transitions: If the DHCP address of the LAN interface comes up this expression only changes to true after 10 seconds. If the DHCP address goes down, this change is propagated immediately to the created variable *ip.ctx:ROUTER.if:LAN.addr:DHCP:UP-STABLE*.

```
node(cfg)#for ip.ctx:ROUTER.if:LAN.addr:DHCP \
    expression UP-STABLE \
        DEBOUNCEINC(up, 10s)
```

Example: Combining link states of all addresses of an interface: The following expression creates a new variable *ip.ctx:ROUTER.if:LAN.ALL-ADDR-UP*, which only evaluates to true if the link state of all addresses of the LAN interface is up.

```
node(cfg)#for ip.ctx:ROUTER.if:LAN \
    expression ALL-ADDR-UP \
        AND(addr:%.up)
```

Example: Wait until the system is up and ready for communication: The following expression creates a new variable *READY*, which only evaluates to true communication with the device can start. This is the case if the system reports up and if the WAN link is up for a minute and if NTP time synchronization has been taken place.

```
node(cfg)#expression READY \
    "sys.up \
    && DEBOUNCEINC(ip.ctx:ROUTER.if:WAN.up, 1m) \
    && ntp.sync"
```

Example: Recognize longer down-cycles: The following expression creates a new variable *ip.ctx:ROUTER.if:LAN.LONG-DOWN* that changes to true if the link state of the LAN interface is down for more than ten minutes.

```
node(cfg)#for ip.ctx:ROUTER.if:LAN \
    expression LONG-DOWN \
        "!up && STABLE(up, 10m)"
```

The *STABLE(up, 10m)* term switches to true if the *ip.ctx:ROUTER.if:LAN.up* variable stays at the same value (true or false) for ten minutes. This has to be combined with the term *!up* such that the resulting variable only evaluates to true if the link is down for that period of time.

State Profiles

State profiles build simple state machines from system variables and expressions. These state machines are used by other applications such as the SIP user agent. SIP drops incoming packets when the system is overloaded (see Chapter 51, "[SIP Overload Configuration](#)" on page 638). The overload situation is detected by the *OVERLOAD* state profile, which is fully programmable by the user.

Consider the example depicted in [figure 20](#) on page 178:

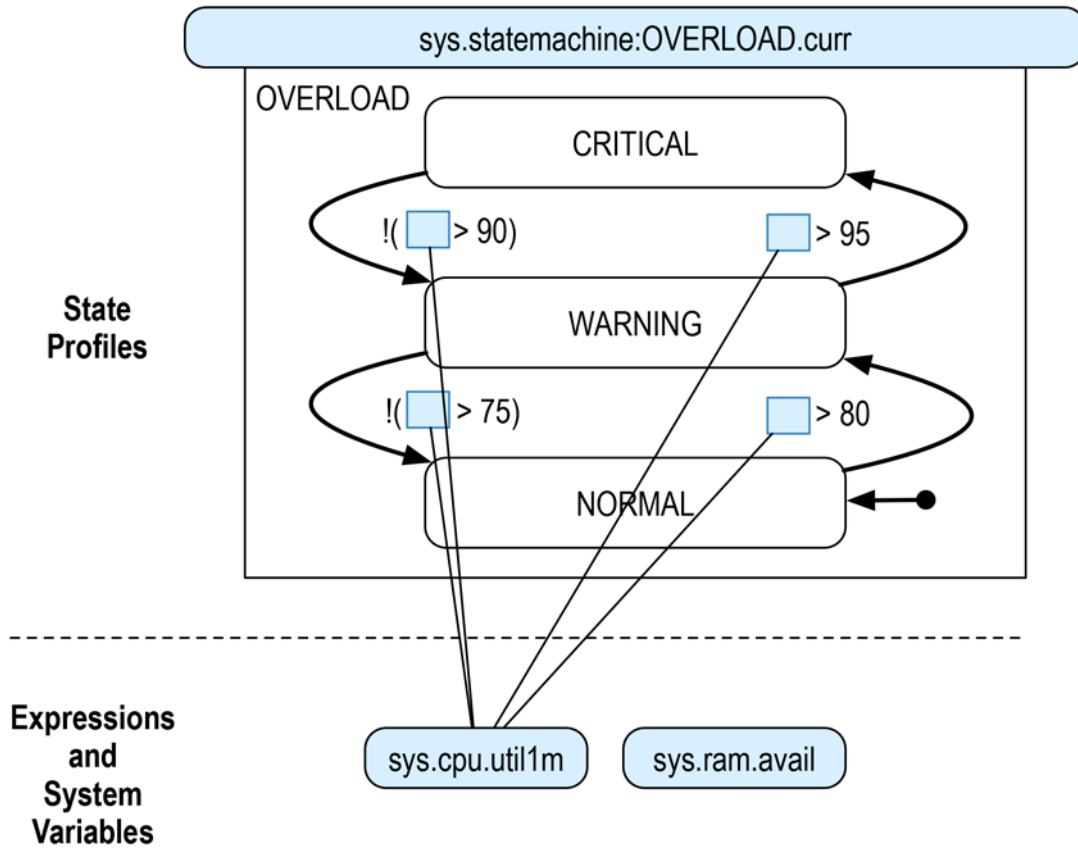


Figure 20. State Profile

Next to the initialization state *NORMAL* the state profile *OVERLOAD* defines two additional states, *WARNING* and *CRITICAL*. The transitions between those states are specified by expressions using the system variable `sys.cpu.util1m`, which exposes the CPU utilization, averaged over one minute. If the CPU utilization raises above 80% the *OVERLOAD* state machine changes to the *WARNING* state, if it crosses the 95% margin the *CRITICAL* state is entered. You can also define hold conditions to build a hysteresis as depicted in [figure 21](#) (green=*NORMAL*, yellow=*WARNING*, red=*CRITICAL*).

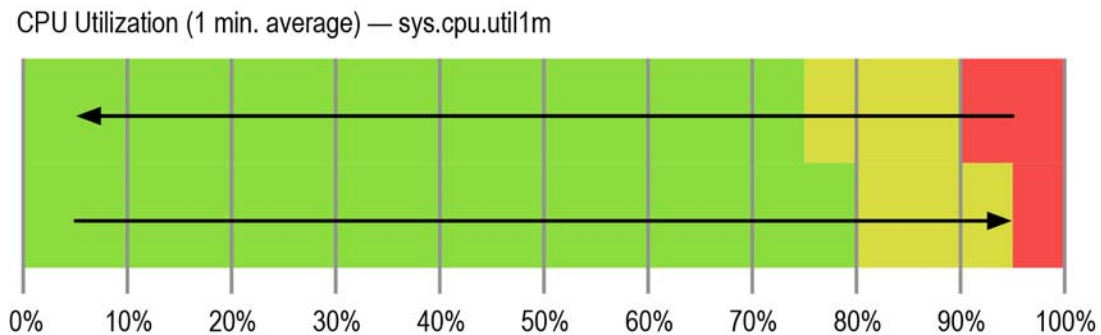


Figure 21. State Transition Hysteresis

In the example above the state profile only changes back from the CRITICAL to the WARNING state if the CPU utilization drops to 90% or below. This example is configured by the CLI commands below:

```
profile state OVERLOAD
  init-state NORMAL

  state 1 WARNING
    enter-if sys.cpu.util1m>80 hold-when sys.cpc.util1m>75

  state 2 CRITICAL
    enter-if sys.cpu.util1m>95 hold-when sys.cpu.util1m>90
```

A state profile exposes its state as another system variable. For example, the *OVERLOAD* profile defines the variable *sys.statemachine:OVERLOAD.curr*, a string variable that is either set to 'NORMAL', 'WARNING', or 'CRITICAL', dependent on the current state of the state machine. Other components may use this variable to react on state changes.'

Default OVERLOAD state profile

The device automatically creates a default OVERLOAD state profile, populated with reasonable expressions to switch the device into WARNING or CRITICAL state. Other applications such as the SIP user agent uses this OVERLOAD profile to limit VoIP calls if the system resources are in a critical state. The default CLI configuration for the OVERLOAD state profile is printed in the code snippet above

The default OVERLOAD state profile transitions to state

- **WARNING:** if the CPU utilization (1 min. av.) rises above 80%.
- **CRITICAL:** if the CPU utilization (1min. av.) rises above 95%.

The device should not reach any of those thresholds if it is operated under nominal load as described in the product manual. However, if the device is under a denial-of-service attack, using the OVERLOAD state profile to define the SIP overload behavior makes sure the ongoing calls are not affected by an excessive load (see Chapter 11, "[Programmable System-Event Configuration](#)" on page 149).

Create and configure a new state profile

The following procedure shows how to create a state profile similar to the default OVERLOAD profile shown in the example above. The procedure illustrates how to create a new profile, configure the name of the initialization state, create a new state, and add enter and hold conditions for that state. You may repeat steps 3 and/or 4 to create multiple states or to add multiple conditions to a state.

Mode: Configure

Step	Command	Purpose
1	<i>node(cfg)#profile state name</i>	Creates a new state profile called name. Use the same name when linking another component to the profile with the use profile state command.
2	<i>node(pf-state)[name]#init-state init-state</i>	Sets the profile's initialization state name. The state machine is in this state if none the entering-conditions of any other state evaluates to true. If you don't enter this command the default init state will be called INIT .

Step	Command	Purpose
3	<code>node(pf-state)[name]#state state</code>	Appends a new state called <i>state</i> in the state profile <i>name</i> . As long as no entering conditions are configured for the state the state machine will never enter the state.
4	<code>node(state)[name]#enter-if enter-expression [hold-when hold-expression]</code>	Configures an <i>enter-expression</i> that if it evaluates to true will cause the state machine to enter the current state. To build a hysteresis, use the optional <i>hold-expression</i> , which defines how long the state machine should stay in this state. If absent the <i>hold-expression</i> will be set to the <i>enter-expression</i> .

Note States within a state profile are ordered by an ordinal number, e.g. **state 1 WARNING, state 2 CRITICAL**. This order is important. The state machine tries to enter the state with the highest ordinal number the enter-expression of which evaluates to true. Consider again Figure 5: If the CPU utilization changes to 98% the enter expression of both states, WARNING and CRITICAL are true. Since CRITICAL is a higher-order state than WARNING the state machine immediately enters the CRITICAL state.

Check the configuration of state profiles

Use the following show command to either get a list of all configured state profiles or to print detailed information about one particular profile.

Mode: Operator exec

Step	Command	Purpose
1	<code>node>show profile state [name]</code>	If a profile <i>name</i> is specified this command prints detailed information about the state profiles. If the profile <i>name</i> is skipped, a list of all profiles is printed.

Example: Print a list of all state profiles:

```
node>show profile state
State Profiles
=====

State Profile  Current State
-----
LED-STATE     UP
NTP-STATE     INIT
OVERLOAD      WARNING
```

Example: Print detailed information about the OVERLOAD state profile:

```
State Profile: OVERLOAD
=====

Init State:          NORMAL
Other States:        CRITICAL, WARNING
Current State:       NORMAL (init state)
System-Variable Name: sys.statemachine:OVERLOAD.curr
```


Debug state transitions

Use the following debug command to observe state transitions:

Mode: Operator exec

Step	Command	Purpose
1	<code>node>debug state full-detail</code>	Enables the real-time debugger that prints information about state transitions of state profiles to the current terminal.
2	<code>node>debug variable detail 1</code>	Additionally observes changes of system variables, on which the state-transition expressions are built.

Example: Observe all state-machine transitions and changes of all system variables:

```
node>debug state full-detail
node>debug variable detail 1
15:01:23.000 VAR      # [sys.cpu.utillm] 94 -> 98
15:01:23.000 STATE   # [OVERLOAD] Re-evaluating state-machine...
15:01:23.000 STATE   # [OVERLOAD] Check state 1 WARNING:
15:01:23.000 STATE   # [OVERLOAD] Enter condition: true
15:01:23.000 STATE   # [OVERLOAD] Hold condition: true
15:01:23.000 STATE   # [OVERLOAD] Check state 2 CRITICAL:
15:01:23.000 STATE   # [OVERLOAD] Enter condition: false
15:01:23.000 STATE   # [OVERLOAD] Hold condition: false
15:01:23.000 STATE   # [OVERLOAD] Transition to state WARNING
```

Example: To test whether the system behaves as expected in a state that is rarely entered you may temporarily add a new expression to that state that always evaluates to true.

```
node>debug state full-detail
node>debug variable detail 1
node>enable
node#configure
node(cfg)#profile state OVERLOAD
node(pf-state)[OVERLOAD]#state WARNING
node(state)[WARNING]#enter-if true
15:01:23.000 STATE   # [OVERLOAD.WARNING.true] onNew
15:01:23.000 STATE   # [OVERLOAD.WARNING] Update expression...
15:01:23.000 STATE   # [OVERLOAD.WARNING] Enter expression:
    BOOL(sys.cpu.utillm >= 80) || BOOL(true)
15:01:23.000 STATE   # [OVERLOAD.WARNING] Hold expression:
    BOOL(sys.cpu.utillm >= 75) || BOOL(true)
15:01:23.000 STATE   # [OVERLOAD] Re-evaluating state-machine...
15:01:23.000 STATE   # [OVERLOAD] Check state 1 WARNING:
15:01:23.000 STATE   # [OVERLOAD] Enter condition: true
15:01:23.000 STATE   # [OVERLOAD] Hold condition: true
15:01:23.000 STATE   # [OVERLOAD] Check state 2 CRITICAL:
15:01:23.000 STATE   # [OVERLOAD] Enter condition: false
15:01:23.000 STATE   # [OVERLOAD] Hold condition: false
15:01:23.000 STATE   # [OVERLOAD] Transition to state WARNING
```

Chapter 12 Alarm Management

Chapter contents

- Introduction 183
- Alarm Configuration Task List 183
 - Viewing alarms 183
 - Changing alarm options 184

Introduction

This management component is responsible for managing the alarm sub-system. It allows you to add/remove and see alarms. Trinity is equipped with alarm sub-system. Any component may register alarms and once triggered they will show up in the alarms table. Each alarms can have one of the following severities:

- **Informative:** General informative alarm
- **Minor:** Minor warning like alarm. The system is about to overheat
- **Major:** System is above the allowed heat threshold
- **Critical:** System had a catastrophic failure

Alarms are predefined and are configured by the trinity. You cannot add alarms dynamically. However you can change the severities and you can enable and disable them.

Alarm Configuration Task List

The following sections describe how to configure/use the Alarm component:

- Changing alarm options

Viewing alarms

Mode: administrative access

Step	Command	Purpose
1	<code>[device]#show alarms</code>	Lists alarms and their configurations

Example output:

```
00A0BA0~(cfg)#show alarms
```

#	Description	Severity	Count	Alarm Status	Fault Status	Auto Clear
0	CPU Over Threshold	Major	0			no
1	DSL 0/0	Major	0			no
2	DSL 0/1	Major	0			no
3	DSL 0/2	Major	0			no
4	DSL 0/3	Major	0			no
5	Hardware Sensor Failure	Major	0			no
6	Temperature Over Threshold	Major	0			no

Fields:

- **Count** – Alarm counter. Shows how many time this specific alarm has been triggered.
- **Fault status** – If true, signifies that there is a fault present. System is malfunctioning
- **Alarm status** – If true, signifies that the alarm is active regardless of the fault status. You can clear the alarm if the fault status is no longer present.
- **Auto Clear** – a flag that tell the system to clear the alarm once the fault condition has gone.

Changing alarm options

Mode: administrative access

Step	Command	Purpose
1	<code>[device]#alarm <name> clear</code>	This command enables and disables the clear-all flag of the specified alarm.
2	<code>[device]#alarm clear-all</code>	This command enables and disables the clear-all flag of all alarms present.

Chapter 13 **Auto Provisioning of Firmware and Configuration**

Chapter contents

Introduction	186
Provisioning Profile	186
Creation	186
Destination	186
Destination Script	186
Destination Configuration	187
Destination Upload	187
Locations	188
Placeholders in Locations	188
Conditional Placeholders in Locations	189
Activation	190
Include Command	190
User authentication	191
Server authentication	191
Client Authentication	191
Private Key	191
Own Certificate Chain	191
Disable Client Authentication	192
TLS Profile commands used from provisioning	192
TR-069	194
PKI commands used from provisioning	195
Using Provisioning	196
Provisioning Reset	196
Provisioning Status	196

Introduction

This chapter provides an overview of Trinity's Auto Provisioning capabilities and tasks involved to configure it.

The auto provisioning capability enables you to automatically distribute up-to-date configurations and firmware to a large number of units using TFTP, HTTP, or HTTPS. It works as follows: The unit downloads a specific file from a TFTP server. If this file has changed since the last download, it is stored and executed. If the file on the server did not change since the last download, no action is taken. If the units are configured to do auto provisioning, a network operator can only update the firmware files on the server, which automatically distributes it to all units. The "profile provisioning" configures this.

Provisioning Profile

Creation

A provisioning profile is a set of commands dedicated for keeping updated a specific file or software. For example one profile can keep the firmware updated and another profile can keep the configuration up to date.

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg)# [no] profile provisioning <name></code>	Creates or enters a new provisioning profile with the according name and enters it for editing.

Destination

The destination specifies the type of file which is keeping up to date. There are six possible destinations available:

- **script:** Used for firmware update and firmware update on daughter boards.
- **configuration:** Used for updating the startup configuration.
- **upload:** Uploads the current configuration back to the server.
- **firmware:** Used for updating the firmware
- **wizard:** Used for provisioning wizard xml files
- **dialplan:<file-name>:** Used for provisioning dial plan files specifying a file name.

Mode: Profile provisioning

Step	Command	Purpose
1	<code>node(pf-prov)[name]# destination (script configuration upload firmware wizard dialplan:<file-name>)</code>	Specifies the type of provisioning

Destination Script

The destination script is used to provision firmware, firmware for daughter boards or updating other specific files on the device, for example WEB pages. Multiple locations can be used and locations for scripts can use the following protocols: TFTP, HTTP and HTTPS. Together with HTTP or HTTPS it is possible to make user

authentication. Using HTTPS it is possible to make server authentication with a certificate. The target file of a location can be either a **.tar** file containing a **manifest** file. Or it can be a manifest file directly.

In the case of a **.tar** file the provisioning needs to download the complete tar file each time the provisioning triggers. For firmware updates this can generate a lot of traffic. Due to when the manifest in the tar is detected being the same file as the last updated, the provisioning does not perform an update. The advantage of using a **.tar** directly is that the delivered files can be used as received without additional manipulations.

In the case of a **manifest** the provisioning downloads only the manifest file each time the provisioning is triggered. If then a change compared to the last downloaded manifest is detected all other files belonging to the delivery are downloaded. Therefore these files must be stored exactly at the same place as the manifest file on the server.

In either case the activation reload immediate should be configured to bring new software into actions as soon as possible.

An example configuration for provisioning software from a **.tar** file:

```
profile provisioning FIRMWARE
  destination script
  use profile tls DEFAULT
  location 1 tftp://firmware.patton.com/devices/SN5300.tar
  location 2 tftp://192.168.2.2/devices/SN5300.tar
  activation reload immediate
```

An example configuration for provisioning software from a manifest file:

```
profile provisioning FIRMWARE
  destination script
  use profile tls DEFAULT
  location 1 tftp://firmware.patton.com/devices/SN5300/manifest
  location 2 tftp://192.168.2.2/devices/SN5300/manifest
  activation reload immediate
```

Destination Configuration

The destination configuration is used to provision the startup configuration of the device. Multiple locations can be used and locations for configuration can use the following protocols: TFTP, HTTP and HTTPS. Together with HTTP or HTTPS it is possible to make user authentication. Together with HTTPS it is possible to make server authentication with a certificate.

The activation reload immediate should be configured to bring new configuration into action as soon as possible.

Destination Upload

The destination upload is used to upload the startup configuration from the device up to a server. Only one location can be used and only with the protocol TFTP. There should be no activation configured because there is no update on the device itself.

- **Destination Firmware:** The destination firmware is used to provision newer firmware to the device
- **Destination Wizard:** The destination wizard allows for provisioning of wizard xml files

Locations

Under locations the protocol, server and the file on that server is specified. Some of the features of provisioning are only available for certain protocols. The available protocols are:

- **TFTP:** Standard download protocol without security features, can be used for uploads too.
- **HTTP:** Web download with the possibility of requesting digest authentication from the device. Not available for uploads.
- **HTTPS:** Secure web download with the possibility of requesting digest authentication from the device and the possibility of authenticate the certificate of the webserver. Not available for uploads.

At least one location has to be configured for a profile. But it is possible to configure multiple locations as fallback if the first locations cannot be reached. Important is that on all the locations in the same profile the exact same file has to be reached. Updating or replacing of these files, should be done simultaneously.

Mode: Profile provisioning

Step	Command	Purpose
1	<code>node(pf-prov)[name]# location [<index>] tftp://server/path/file</code>	Adds a location with tftp protocol at the index or at the end
2	<code>node(pf-prov)[name]# location [<index>] http://server/path/file</code>	Adds a location with http protocol at the index or at the end
3	<code>node(pf-prov)[name]# location [<index>] https://server/path/file</code>	Adds a location with https protocol at the index or at the end
4	<code>node(pf-prov)[name]# location [<index>] \$(dhcp.66)</code>	Adds a location with tftp server from the DHCP option 66
5	<code>node(pf-prov)[name]# location [<index>] \$(dhcp.66)<suffix></code>	Adds a location with tftp server from the DHCP option 66
6	<code>node(pf-prov)[name]# no location <index></code>	Removes the location at the specified index

Placeholders in Locations

In the server, path and filename of the location placeholders can be used. These placeholders are replaced by their corresponding value and can be used to build the provisioning server location request. The following placeholders are available:

- **\$(cli.major)**— CLI major version as numerical value
- **\$(cli.minor)**— CLI minor version as numerical value
- **\$(system.hw.major)**— Hardware major version as numerical value
- **\$(system.hw.minor)**— Hardware minor version as numerical value
- **\$(system.sw.major)**— Software major version as numerical value
- **\$(system.sw.minor)**— Software minor version as numerical value
- **\$(system.sw.build)**— Software build version as numerical value
- **\$(system.sw.date)**— Software release date as string in the format 'YYYY-MM-DD'

- **\$(system.product.name)**— Product name as string
- **\$(system.mac)**—MAC address of ETH 0/0 as string in the format ‘AABBCCDDEEFF’ (without “:” between the hexadecimal characters)
- **\$(system.serial)**—serial number of the unit as string in the format ‘AABBCCDDEEFF’
- **\$(dhcp.66)**—TFTP server provided from DHCP. The following fields in the DHCP offer are considered in descending priority:
 - **option 66:** TFTP server IP, as string delivered by the DHCP server.
 - **siaddr:** BOOTP next server, converted to string delivered by the DHCP server.
 - **sname:** BOOTP server name, as string delivered by the DHCP server.
 - When no DHCP is enabled or none of the above fields is present, the string is empty.
- **\$(dhcp.67)**—TFTP file name provided form DHCP. The following fields in the DHCP offer are considered in descending priority:
 - **option 67:** TFTP file name, as string delivered by the DHCP server.
 - **file:** BOOTP file name, as string delivered by the DHCP server.
 - When no DHCP is enabled or none of the above fields is present, the string is empty.

Conditional Placeholders in Locations

A conditional placeholder may also be used to choose between two values and or placeholders in the function of a condition. It has the following syntax:

- **\$(number_1=number_2|true_placeholder|false_placeholder)**
- **\$(number_1>number_2|true_placeholder|false_placeholder)**
- **\$(number_1>=number_2|true_placeholder|false_placeholder)**
- **\$(number_1<number_2|true_placeholder|false_placeholder)**
- **\$(number_1<=number_2|true_placeholder|false_placeholder)**

Explanation of elements:

- **number_1 and number_2:** Values used to determine if the condition is true or false. Can be a placeholder corresponding to a numerical value or any numerical value.
- **true_placeholder:** String to replace the condition if the condition is true. This can be also a nested placeholder or nested conditional placeholder.
- **false_placeholder:** String to replace the condition if the condition is false. This can be also a nested placeholder or nested conditional placeholder. This can be left out and if the condition is false then the condition is replaced by an empty string

Here are some examples for locations with conditional placeholders:

- `tftp://192.168.1.1/software/${$(system.hw.major)>=2|new|old}/image.tar`
- `tftp://192.168.1.1/software/${$(system.sw.major)=3|$(system.product.name)|image}.tar`
- `tftp://192.168.1.1/software/${$(cli.major)>=4|trinity}/image.tar`

Activation

For the destination script and configuration it is preferred to let the device reboot after the update in order for the update to become active. The following modes are available:

- **activation reload immediate:** After an update of the firmware or the configuration a reload happens immediately to bring the updated files into action. In most cases this should be used.
- **activation reload deferred:** The reload is not triggered automatically and happens later only when executing the command “reload if-needed”.
- **no activation:** No reload is executed after the provisioning. This should be used only for the upload destination.
- **graceful:** Activates the provisioned files or firmware with a reload as soon as there are no ongoing calls on the device any more.

Mode: Profile provisioning

Step	Command	Purpose
1	node(pf-prov)[name]# [no] activation reload (immediate deferred graceful)	Specifies if and when an activation with a reload has to take place, to bring the updated files into action.

Include Command

The include command enables the user to execute an additional, separate configuration file (i.e. a generated configuration via wizard or a manually created one) within an existing configuration, allowing nested configurations. You can use the include command interactively on the CLI prompt to execute the commands in the file immediately or put it inside a startup configuration to execute another configuration file at boot-up. The include command can execute any command that can already be executed through the CLI.

Example: Add a provisioning profile and a timer, defined in a separate configuration file, into the startup configuration:

Existing configuration on the Patton device (example.cfg):

```
profile provisioning PF_PROVISIONING_CONFIG
destination configuration
location 1 tftp://myserver.com/prov-configuration timer PROVISIONING now + 3 minutes "provisioning execute PF_PROVISIONING_CONFIG"
```

Using the include command via the CLI or in the startup configuration will create a new provisioning profile and a new timer:

```
# commands before include
include config:example.cfg
# commands after include
```

Note The commands in the included configuration file are executed after the commands before the include statement and before the commands after the include statement. That is, if there already is a "profile provisioning PF_PROVISIONING_CONFIG" configuration in the startup configura-

tion the configuration in the `example.cfg` will overwrite that. You are able to include multiple configuration files and it is also possible to place include commands in included configuration (recursion).

User authentication

For configuration and firmware provisioning through HTTP or HTTPS optional authentication is available. It is required if the final server where the configured items have to be downloaded is configured for basic or digest authentication. This has no impact when the TFTP protocol is used.

Mode: Profile provisioning

Step	Command	Purpose
1	<code>node(pf-prov)[name]# [no] authentication [realm <realm>] username <user> (no-password password <password>)</code>	Adds/Removes authentication credentials for the configured realm. If no realm is given the configured credentials act as wild-card. The wild-card is considered if the realm provided in 401 Unauthorized doesn't match any explicit configured realm.

Server authentication

For HTTPS locations, server authentication can be enabled. To do so, a TLS profile has to be bound to the provisioning profile. Be aware that in the case of HTTPS only a subset of the TLS configuration options are used. Here a list of the actual TLS parameters used in the case of HTTPS:

- authentication outgoing
- trusted-certificate

The used TLS profile can be configured as indicated below.

Mode: Profile provisioning

Step	Command	Purpose
1	<code>node(pf-prov)[name]#</code>	Chooses the TLS settings to use for HTTPS locations

Client Authentication

Patton devices are now able to authenticate themselves to the HTTPS server during provisioning (client authentication). The necessary parameters are configured in the TLS profile:

- private key
- own certificate chain

Private Key

A private key file actually contains a private/public key pair. TLS requires them to secure its key exchange. The private key must match the key information included in the first own-certificate.

Own Certificate Chain

Our own certificate is used to identify our device to the remote party when making or accepting TLS

connections. The chain of own certificates consists of an ordered list of one or more links to stored certificates. The first certificate in the list (index 1) is our public key that is sent to the remote TLS device. It is important that the private key linked by the TLS profile corresponds to the first public key in this certificate. Therefore it is mandatory to configure a link to a private key on the TLS profile.

If you want to use a self-signed certificate, the chain of own certificates only contains one entry - a link to that certificate. Otherwise, if your certificate is signed by one or more levels of certification authorities, those intermediate certificates have to follow in the chain. That is, the next entry with index 2 links to the certificate of the certification authority that signed our own certificate at index 1. The entry with index 3 links to the certificate of the next higher-level certification authority, and so on. The certificate of the root certification authority is not required and may be omitted or may be configured as the final entry in the list.

Disable Client Authentication

If either the private-key or the own-certificate-chain is not configured or not valid, client authentication is disabled. A HTTPS connection to the configured provisioning location can still be established. If the server does not authenticate the client, the provisioning still performs the configured action. However, if the server requests the device to authenticate itself, this authentication fails and the server terminates the connection. In this case the provisioning continues with the next location in the provisioning profile.

Provisioning profile commands used:

Mode: profile provisioning <name>

Step	Command	Purpose
1	node(pf-prov)[<name>]# [no] use profile tls <tls-profile-name>	Chooses the TLS settings for client and server authentication that are applied to all HTTPS locations.

TLS profile commands used for provisioning:

Mode: profile provisioning <name>

Step	Command	Purpose
1	node(pf-tls)[<name>]# [no] private-key pki:private-key/key	Configures the private (and public) key to be used for the TLS key-exchange encryption.

Mode: profile tls <name>

Step	Command	Purpose
1	node(pf-tls)[<name>]# [no] own-certificate [index] pki:own-certificate/certificate	Configures the chain of own certificates. Inserts the certificate at the specified index into the chain or appends it at the end if no index is specified. The sequence of own certificates is of importance!

TLS Profile commands used from provisioning

To enable the server authentication the following command in the TLS profile can be used. Enabling server

authentication makes use of the configured trusted certificates from the TLS profile to verify the trustworthiness of the server certificate. If the server authentication is disabled the server certificate is accepted always.

Mode: Profile TLS

Step	Command	Purpose
1	node(pf-tls)[name]# [no] authentication outgoing	Enables or disables server authentications with HTTPS provisioning

If server authentication is enabled the trusted-certificate command in the TLS profiles defines which are the root certificates we trust, and from one of these the server certificate must be signed off.

There are following variants to configure:

- **built-in-defaults:** Use all certificates which are built in provided from the software as trusted. This group of certificates is dependent of the software which is running on the device. With each software update this whole list is updated too. New certificates may be added. Certificates are removed only if they expire or needs to be considered as not trustworthy anymore. The user cannot make any changes to this group.
- **user-stored:** Use all certificates which are installed from the user. It is possible to install certificates from tftp or from the built-in-defaults certificate group.
- **all-stored:** Use all built-in-defaults certificates and all user-stored certificates as trusted certificates.
- **specific list:** The user can create the list of trusted certificates on his own. Therefore any certificate of the group built-in-defaults or user-stored can be used. All certificates which are not listed are not considered as trusted certificates.

Mode: Profile tls

Step	Command	Purpose
1	node(pf-tls)[name]# trusted-certificate (all-stored built-indefaults user-stored)	Configures to use all imported trusted certificates for the current TLS profile. This is the default setting.

Mode: Profile tls

Step	Command	Purpose
1	node(pf-tls)[name]# [no] trustedcertificate pki:trustedcertificate/ certificate	Adds or removes a certificate to or from the set of trusted certificates of the current PKI profile. Removing the last link sets the TLS-profile configuration back to the default setting, which is to use all trusted certificates in PKI.

Mode: Profile tls

Step	Command	Purpose
1	node(pf-tls)[name]# [no] trustedcertificate pki:trusted-certificatedefaults/ certificate	Adds or removes a certificate to or from the set of trusted certificates of the current PKI profile. Removing the last link sets the TLS-profile configuration back to the default setting, which is to use all trusted certificates in PKI.

TR-069

SmartNode devices now support the TR-069 standard. The data model is based on Device:2.7 (TR-181 Issue 2 Amendment 7), but currently only supports the nodes *DeviceInfo* and *ManagementServer*. The following features are supported:

- Auto-Configuration Server (ACS) discovery (DHCP Option 43 or 125)
- Software/Firmware image management
- Configuration-File management (download and upload)
- Upload of log reports file
- Factory reset RPC
- Reboot RPC
- HTTP connection request
- UDP connection request (connection request via NAT gateway using STUN sever: TR-069 Annex G, formerly TR-111)

Mode: configure

Step	Command	Purpose
1	<code>node(cwmp-client)# [no reset] acs <url></code>	Sets the URL of the Auto-Configuration Server (ACS).
2	<code>node(cwmp-client)# [no reset] authentication username <user> [password <password>]</code>	Sets the credentials used to connect to the ACS.
3	<code>node(cwmp-client)# [no reset] user profile tls <profile></code>	Applies a TLS profile for server and/or client authentication when establishing an HTTPS connection to the ACS.
4	<code>node(cwmp-client)# [no reset] discovery</code>	Enables/disables ACS discovery via DHCP option 43 or 125. Your DHCP server has to be configured accordingly.
5	<code>node(cwmp-client)# [reset] provisioning-code <code></code>	Sets the TR-069 provisioning code (see TR-069 and TR-181).
6	<code>node(cwmp-client)# [no reset] bind ipaddress <interface> <label></code>	Selects the local IP address used for CWMP.
7	<code>node(cwmp-client)# [reset] connection-request username <user> [password <password>]</code>	Sets the credentials used by ACS to make HTTP connection requests.
8	<code>node(cwmp-client)# [no reset] periodic-inform interval <seconds> [time <time>]</code>	Enables/disables periodic inform (see TR-069).
9	<code>node(cwmp-client)# [no reset] session-retry-maximum <></code>	Sets the maximum CWMP session retry.
10	<code>node(cwmp-client)# [no reset] session-retry-wait-interval minimum <min> [multiplier <multiplier>]</code>	Sets the smallest CWMP session-wait interval and the interval multiplier (see TR-069).
11	<code>node(cwmp-client)# [no reset] http-compression {gzip deflate}</code>	Enables/disables HTTP compression.

Step	Command	Purpose
12	node(cwmp-client)# [no] shutdown	Enables/disables CWMP.

Mode: cwmp-client / stun

Step	Command	Purpose
1	node(stun)# [reset] stun	Enters the STUN configuration mode to configure Session Traversal Utilities for NAT. STUN is used by the UDP-connection-request feature.
2	node(stun)# [no reset] server <host ip> [port <port>]	Sets the STUN server host-name or IP address (default port 3478).
3	node(stun)# [no reset] authentication user-name <user> [password <password>]	Sets the credentials used to connect to the STUN server.
4	node(stun)# [no reset] keep-alive-period minimum <min> [maximum <max>]	Configures the STUN keep-alive-window period (see TR-181–Issue 2–Amendment 7).
5	node(stun)# [no] shutdown	Enables/disables STUN.

Executable commands:

Command	Purpose
node# show cwmp-client [detail <detail> full-detail] [continuously]	Shows CWMP statistics

PKI commands used from provisioning

Under PKI the group of built-in-defaults trusted certificates is used. This group is managed from Patton and can be changed only through a software update. This group of certificates is dependent of the software which is running on the device. With each software update this whole list is updated too. New certificates may be added. Certificates are removed only if they expire or need to be considered as not trustworthy anymore. The user cannot make any changes to this group. But it is possible to copy certificates from this group to another group or to user-stored certificates or to a tftp server for the usage on other devices.

Mode: configure

Step	Command	Purpose
1	node# copy pki:/trustedcertificates- defaults/<certificate> pki:/trusted-sertificates/<certificate>	Copy a certificate from the built-in-defaults group to the user-stored group to be used there

Mode: Configure

Step	Command	Purpose
1	node# copy pki:/trustedcertificates-defaults/<certificate> tftp://<server/path/certificate>	Copy a certificate from the built-in-defaults group to a tftp server to be installed on other devices

Using Provisioning

To trigger a provisioning profile the execute command has to be used.

Mode: configure

Step	Command	Purpose
1	node# provisioning execute <name>	Executes the provisioning profile immediately one time

To continuously poll for firmware or configuration changes, use the provisioning execute command together with the timer command. Here's how to do both firmware and configuration provisioning, with a polling interval of one day:

```
timer FIRMWARE_UPDATE now + 2 minutes every day "provisioning execute FIRMWARE"
timer CONFIG_UPDATE now + 2 minutes every day "provisioning execute CONFIG"
```

Provisioning Reset

When a device executes a provisioning of a script successfully, it stores the information about executing that script permanently on the device. This is to avoid repeated execution of the same script. Now, if the user manually performs software updates or switching of images and expects the provisioning to execute the software update script after that, this is never going to happen because of the stored state of the script. Therefore a command is introduced to manually reset the stored provisioning state and let the provisioning execute the next time it is triggered.

Mode: Enable

Step	Command	Purpose
1	node#provisioning reset [<profile-Name>]	Resets the stored provisioning state for all the provisioning profiles or only the one specified with <i>profileName</i> .

Provisioning Status

To show the status of provisioning events, the following can be done to view this information.

Mode: Operator

Step	Command	Purpose
1	node>show provisioning status [continuously]	Shows current and pending execution of provisioning profile

Chapter 14 **Ethernet Port Configuration**

Chapter contents

Introduction	198
Ethernet Port Configuration Task List	198
Entering the Ethernet Port Configuration Mode	198
Configuring Medium for an Ethernet Port	198
Binding an Ethernet Port	199
Multiple IP Addresses on Ethernet Ports	200
Configuring a VLAN	201
Configuring Layer-2-CoS to Service-class Mapping for a VLAN	202
Closing an Ethernet Port	203

Introduction

This chapter provides an overview of Ethernet ports and describes the tasks involved in configuring Ethernet ports through the Trinity operating system.

Ethernet Port Configuration Task List

To configure Ethernet ports, perform the tasks described in the following sections. Most of the tasks are required to have an operable Ethernet port, some of the tasks are optional, but might be required for your application.

- Entering the Ethernet port configuration mode (see page 198)
- Configuring medium for an Ethernet port (see page 198)
- Configuring Ethernet encapsulation type for an Ethernet port (see page 199)
- Binding an Ethernet port to an IP interface (see page 199)
- Configuring multiple IP addresses on the Ethernet ports (see page 200)
- Configuring a VLAN (see page 201)
- Configuring layer 2 CoS to service-class mapping for an Ethernet port (advanced) (see page 202)
- Closing an Ethernet port (see page 203)

Entering the Ethernet Port Configuration Mode

To enter port configuration mode and begin configuring an Ethernet port, enter the command **port ethernet slot port** in administrator execution mode. The keywords *slot* and *port* represent the number of the respective physical entity.

Configuring Medium for an Ethernet Port

All Ethernet ports are configured by default to auto-sense both the port speed and the duplex mode. This is the recommended configuration. Command options are (if supported by the platform):

- **auto**—auto-negotiate the port speed and duplex, advertising all supported speeds and duplexes.
- **negotiated**—auto-negotiate the port speed and duplex, only advertising the specified speed and duplex
- **manual**—disable auto-negotiation and force the port speed and duplex. The remote port must also be configured to force speed and duplex. Otherwise, the remote port will operate in half-duplex, resulting in collisions.
- **10**—for 10 Mbps
- **100**—for 100 Mbps
- **1000**—for Gigabit Ethernet
- **half**—for half-duplex
- **full**—for full-duplex
- **sfp**—for ports that support both copper and SFP modules, force the port to use the SFP module

This procedure describes how to configure the medium for the Ethernet port on *slot* and *port*

Mode: Configure

Step	Command	Purpose
1	<code>device(cfg)#port ethernet slot port</code>	Enters the Ethernet port configuration mode for the physical Ethernet connector on <i>slot</i> and <i>port</i> .
2	<code>device(prt-eth)[slot/port]#medium { auto negotiated speed duplex manual speed duplex [stp] }</code>	Configures the Ethernet port's medium.

Example: Configuring medium for an Ethernet port

The following example shows how to configure medium auto-sense for the Ethernet port on slot 0 and port 0.

```
device(cfg)#port ethernet 0 0
device(prt-eth)[0/0]#medium auto
```

Binding an Ethernet Port

You must bind the Ethernet port to an existing bridge-group, switch-group, and/or interface. When executing the **bind** command, the requested bridge-group, switch-group, and/or interface must exist (see Chapter 22, “[IP Interface Configuration](#)” on page 289 to learn how to create a new IP). If no IP context is given, the system looks for the interface in the default IP context known as *ROUTER*.

Figure 22 shows the logical binding of the Ethernet port at slot 0 on port 0 to the IP interface *LAN*, which is defined in the default IP context *ROUTER*.

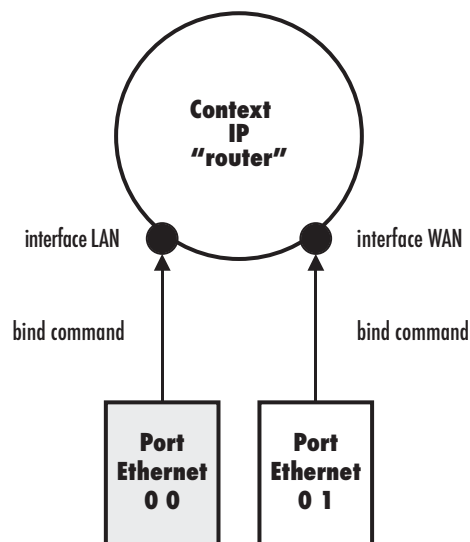


Figure 22. Binding of an Ethernet port to an IP interface

This following procedure describes how to bind the Ethernet port to an already existing IP interface

Mode: Configure

Step	Command	Purpose
1	<code>device(cfg)#port ethernet slot port</code>	Enters Ethernet port configuration mode for the physical Ethernet connector on <i>slot</i> and <i>port</i>
2	<code>node(prt-eth)[0/0]#bind interface [<ctxName>] <ifName></code>	Binds the Ethernet port to the logical IP interface ifname in IP context ctxName

The <ifName> parameter specifies the name of the IP interface within the IP context given with the <ctxName> parameter.

Note Note: The <ctxName> parameter is optional. If you don't provide it, Trinity binds to an IP interface in the default ROUTER context.

Example: Create an IP interface with one static address and bind it to port ethernet.

```
node>enable
node#configure
node(cfg)#context ip ROUTER
node(ctx-ip)[ROUTER]#ip interface LAN
node(ip-if)[ROUTER.LAN]#ipaddress STATIC 10.1.1.1/24
node(ip-if)[ROUTER.LAN]#port ethernet 0 0
node(prt-eth)[0/0]#bind interface ROUTER LAN
node(prt-eth)[0/0]#no shutdown
```

Multiple IP Addresses on Ethernet Ports

It is possible to use multiple IP addresses on an Ethernet port by creating multiple IP addresses on the logical (bound) IP interface.

Note In the previous software releases, multiple IP addresses on an Ethernet port could be achieved by binding the same Ethernet port to multiple IP interfaces. This is no longer supported. Instead, you should now create multiple IP addresses on the same IP interface and bind the Ethernet port to that interface.

The procedures below demonstrate how two different IP addresses (potentially in the same network) can be used on an Ethernet port. However, if necessary any number of static IP addresses and an optional DHCP address can be created on an IP interface.

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg)#context ip [ROUTER]</code>	Enter the IP context mode for the default virtual router.
2	<code>device(ctx-ip)[ROUTER]#interface <name></code>	Creates a new IP interface and/or enters the configuration mode on an existing IP interface.
3	<code>device(ip-if)[ROUTER.<name>]#ipaddress [<label>] <address></code>	Creates a new IP address and network mask on the IP interface <name> or changes the IP address and network mask of an existing IP address, identified by its <label>.

The <label> parameter specifies the name of the address. An IP interface may host more than one IP address. The label must be unique within the IP interface; you may re-use the same label in a different IP interface.

Note Note: The label parameter is optional. If you don't provide it, Trinity creates a new IP address for the interface and automatically assigns it a unique label.

Note Note: If you specify DHCP as label, you create a DHCP address. Only one DHCP address can be created per interface.

The <address> parameter specifies the network address and mask size in dotted-decimal format a.b.c.d/m for IPv4 or in the colon format a:b:c::x/m for IPv6. Alternatively, you may enter the network address and full network mask with two consecutive parameters, a.b.c.d e.f.g.h or a:b:c::x e:f:g::y, respectively.

Example: Create an IP interface with one static and a DHCP address and bind it to port ethernet.

```
node>enable
node#configure
node(cfg)#context ip ROUTER
node(ctx-ip)[ROUTER]#ip interface LAN
node(ip-if)[ROUTER.LAN]#ipaddress STATIC 10.1.1.1/24
node(ip-if)[ROUTER.LAN]#ipaddress DHCP
node(ip-if)[ROUTER.LAN]#port ethernet 0 0
node(prt-eth)[0/0]#bind interface LAN
```

Configuring a VLAN

By default, no VLAN ports are configured on an Ethernet port. One or more VLAN ports can be created on each Ethernet port.

You must bind each VLAN port to an existing IP interface which must be distinct from the IP interface bound by the Ethernet port and by other VLANs. When executing the **bind** command, the requested interface must exist (see Chapter 22, “IP Interface Configuration” on page 289 to learn how to create a new IP interface).

For incoming VLAN packets, each of the 8 possible layer-2 class of services (CoS) can be mapped to an internal traffic-class. Unless otherwise configured, all CoS values map to the default traffic-class.

By default all VLAN ports are initially disabled. They can be enabled with the **no shutdown** command. The corresponding Ethernet port must also be enabled for the VLAN port to work. If the Ethernet port is disabled, all associated VLAN ports are also disabled.

When a VLAN port is closed, the IP interface that is bound to this port is also closed. All static routing entries that are using this interface change their state to *invalid* and all dynamic routing entries will be removed from the route table manager.

Mode: Configure

Step	Command	Purpose
1	device(cfg)#port ethernet slot port	Enter Ethernet port configuration mode.
2	device(prt-eth)[slot/port]#vlan id	Create new VLAN port.
3	device(vlan)[eth-slot/port.id]#bind interface [ROUTER] name	Bind the VLAN port to the existing interface <i>name</i> . If no IP context is given, the system looks for the interface in the default IP context known as <i>ROUTER</i> .
4	device(vlan)[eth-slot/port.id]#no shutdown	Activate the VLAN port.
5	device(vlan)[eth-slot/port.id]#exit	Returns to the Ethernet port configuration mode
6	device(prt-eth)[slot/port]#no shutdown	Make sure the hosting Ethernet port is also activated.

Configuring Layer-2-CoS to Service-class Mapping for a VLAN

To enable to transport real-time and delay-sensitive services such as VoIP traffic across the network, Trinity supports the delivery of Quality of Service (QoS) information in the CoS (Class of Service) field of the 802.1pQ header (VLAN header). This is a three-bit field (values 0 to 7). Internally we use traffic-class tags to mark packets belonging to a certain class of service (see Chapter 22, “[IP Interface Configuration](#)” on page 289 for more detail on the concept of traffic-classes).

To define the Class of Service (CoS) to traffic-class mapping and vice-versa, the **map** command in the VLAN configuration mode is used. The following procedure describes how to change layer-2-CoS to traffic-class and the traffic-class to layer-2-CoS mapping.

Mode: vlan

Step	Command	Purpose
1	device(vlan)[eth-slot/port.id]#map cos cos to traffic-class traffic-class	Applies for all inbound packets with the layer-2-CoS field set to <i>cos</i> , which sets the traffic-class tag for those packets to <i>traffic-class</i> . New traffic-classes are created on the fly.
2	device(vlan)[eth-slot/port.id]#map traffic-class traffic-class to cos cos	Applies for all outbound packets with the internal traffic-class tag set to <i>traffic-class</i> , which sets the layer-2-CoS field of those packets to <i>cos</i> .

Example: Adding a mapping-table entry

The following example shows how to add a mapping table entry, which converts a layer 2 class of service value of 2 into a traffic-class tag *VOICE* and vice-versa on VLAN 100 on the Ethernet port on slot 0 and port 1.

```
device>enable
device#configure
device(cfg)#port ethernet 0 1
device(prt-eth)[0/1]#vlan 100
device(vlan)[eth-0/1.100]#map cos 2 to traffic-class VOICE
device(vlan)[eth-0/1.100]#map traffic-class VOICE to cos 2
```

Closing an Ethernet Port

An Ethernet port can be closed with the **shutdown** command. This command also disables and closes the IP interface that is bound to that port. All static routing entries that are using this interface change their state to 'invalid' and all dynamic routing entries will be removed from the route table manager.

This procedure describes how to disable the Ethernet port on *slot* and *port*.

Mode: Configure

Step	Command	Purpose
1	device(cfg)#port ethernet slot port	Enters the Ethernet port configuration mode for the physical Ethernet connector on <i>slot</i> and <i>port</i> .
2	device(prt-eth)[slot/port]#shutdown	Disables Ethernet port on <i>slot</i> and <i>port</i> .

The **no** prefix before the **shutdown** command causes the port to open with the interface to which it is bound.

Example: Disabling an Ethernet port

The following example shows how to disable the Ethernet port on slot 0 and port 1.

```
device(cfg)#port ethernet 0 1
device(prt-eth)[0/1]#shutdown
```

Checking the state of the Ethernet port on slot 0 and port 1 shows that the interface was closed.

```
device(prt-eth)[0/1]#show port ethernet 0 1

Ethernet 0 1
-----

Medium : Auto
Administrative State : Down
Bound IP Interface : ROUTER.WAN
IP Addresses : WAN (static): down
  Configured: 20.1.1.1/24
  Operational: 20.1.1.1/24
DHCP (DHCP): down
Requested: (none)
Operational: (none)
Medium : Not Connected
Operational State : Down
Hardware Address : 00:a0:ba:06:d6:0a
MTU : 1500
```

```
ARP : enabled
Multicast: enabled
Rx Statistics : 0 bytes in 0 packets
0 errors 0 drops, 0 overruns
0 multicast packets
Tx Statistics : 2038 bytes in 13 packets
0 errors 0 drops, 0 collisions 0 carrier errors
```

Moreover the IP interface, which is bound to the Ethernet port on slot 0 and port 0 gets also closed. Checking the state of the IP interface *WAN* indicates this with the Status parameter set to shutdown on the interface and down on all its addresses, respectively.

```
device(prt-eth)[0/1]#show ip interface
```

```
IP Interface Status Address Type Status Configured Operational
-----
WAN shutdown WAN static down 20.1.1.1/24 20.1.1.1/24
DHCP DHCP down (none) (none)
```


Chapter 15 **Configuring a Cellular Modem**

Chapter contents

Introduction.....	206
Supported Cellular Modems.....	206
Virtual Ports vs. Physical Modems	206
Cellular Modem Configuration Task List.....	207
Creating a Virtual Port and Linking It to a Physical Modem	207
Configuring the Cellular Network Parameters	208
Configuring the IP Address	208
Enabling the Cellular Modem	208
Displaying the Cellular Modem's Configuration and Status	209
Debugging the Cellular Modem	210
Troubleshooting.....	210
My cellular modem is not getting an IP address	210
My cellular modem has a static public IP address but I can't access my device using it	211

Introduction

This chapter describes how to configure a cellular modem.

Supported Cellular Modems

Trinity supports several cellular modem USB dongles. [Table 17](#) lists the cellular modems supported by Trinity.

Table 17. Cellular modems supported by Trinity

Modem	Control Protocol	IP Address Type
Huawei E3276	Generic	dhcp
Huawei E3531	Generic	dhcp
Netgear AirCard 340U	MBIM	dhcp
Netgear AirCard 341U	QMI	cellmodem
ZTE MF683	Generic	cellmodem
ZTE MF823	Generic	dhcp

The control protocol is the protocol used internally between the Trinity Device and modem to configure the modem and to query its status. The MBIM and QMI protocols provide full access to the modem's configuration and status, whereas the generic protocol does not. This means that for modems supporting MBIM or QMI, Trinity is able to display the modem's connection status and to invalidate routes to the cellular network when the cellular connection drops, but it is not able to for generic modems.

The IP address type must be known when configuring the cellular modem's IP address. Generic and MBIM modems use DHCP to provide Trinity with an IP address, whereas QMI modems use the QMI protocol directly to provide Trinity with an IP address.

Because several modems support either MBIM or QMI, it is possible that other modems may work when plugged into a Trinity Device; however, Patton Electronics only tests the modems listed above in [table 17](#).

Virtual Ports vs. Physical Modems

In order to understand cellular modem configuration, an important concept to understand is that of virtual and physical ports.

Because a cellular modem USB dongle is hot-pluggable, it has some properties that a fixed port does not have. For example, it may be plugged in after the Trinity Device has booted and applied its startup configuration, or it may be unplugged after it has been configured while the Trinity Device is still running. It is desirable to be able to modify and save a modem's configuration even when it is not plugged in, and then to apply that configuration when it is plugged in. It is also desirable to not automatically erase the modem's configuration when the modem is unplugged.

To provide these properties, Trinity makes a distinction between the virtual port and the physical modem. The virtual port must be explicitly created and removed by the user executing the **[no] port virtual cell-modem port** command. It is stored in the startup configuration and contains the modem's configuration. The physical modem is the actual modem, and Trinity is only aware of it while it is plugged in. As soon as it is unplugged, Trinity automatically removes it.

The virtual port is configured with parameters, such as a USB vendor and product ID or an MDN (mobile director number, i.e. the phone number), that are used to match it to a physical modem when that modem is

plugged in. If the virtual port's parameters match the physical modem's information, the virtual port is linked to the physical modem, and its configuration is applied to the modem.

Note You need to provision the modem correctly on a computer before using it on Trinity.

Cellular Modem Configuration Task List

To configure a cellular modem, perform the tasks described in the following sections:

- “Creating a Virtual Port and Linking It to a Physical Modem”
- “Configuring the Cellular Network Parameters” on page 208
- “Configuring the IP Address” on page 208
- “Enabling the Cellular Modem” on page 208
- “Displaying the Cellular Modem's Configuration and Status” on page 209
- “Debugging the Cellular Modem” on page 210

Creating a Virtual Port and Linking It to a Physical Modem

The first step to configuring a cellular modem is to create a virtual port to store the configuration and to specify which physical modem the configuration belongs to. There are two ways to do this. The first is to use the following commands.

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg)#port virtual cell-modem port</code>	Creates the virtual cellular modem port <i>port</i> and enters its configuration mode. Use the <i>no</i> form of the command to remove it.
2	<code>node(vpvt-cell)[port]#link usb [vendor <i>0xvendor</i> product <i>0xproduct</i>] [mdn <i>mdn</i>] [priority <i>priority</i>]</code>	<p>Specifies the parameters used to match this virtual port to a physical modem.</p> <p>When a physical modem is plugged in, it's the parameters specified in this command will be compared to the physical modem's information. Any parameters not specified will be ignored. If all the specified parameters match, this virtual port may be linked to the physical modem.</p> <p>The parameters that may be specified are the USB vendor and product IDs, and the mobile device number (MDN, i.e. the phone number).</p> <p>In addition, a <i>priority</i> may be specified to be used as a tie-breaker. If more than one virtual port matches the same physical modem, the one with the highest <i>priority</i> will be linked to the physical modem.</p>

The second way to create a virtual port is to insert the physical modem. When no virtual ports match the physical modem, Trinity will automatically create a virtual port that matches the physical modem's vendor and product IDs at priority 1.

Configuring the Cellular Network Parameters

To connect to a cellular network, the cellular network parameters are required, namely the access point name (APN). Use the following command:

Mode: port virtual cell-modem

Step	Command	Purpose
1	node(vpirt-cell)[port]# apn apn	Connect to access point <i>apn</i> .

If the APN is not configured, then Trinity will attempt to use a default APN based on the mobile country code (MCC) and mobile network code (MNC) contained in the international mobile subscriber identity (IMSI) read from the modem's SIM card.

Configuring the IP Address

To configure an IP address on a modem, use the following commands:

Mode: Configure

Step	Command	Purpose
1	node(cfg)# context ip ctx-name	Enters IP context <i>ctx-name</i> 's configuration mode.
2	node(ctx-ip)[<i>ctx-name</i>]# interface if-name	Creates IP interface <i>if-name</i> and enters its configuration mode.
3	node(if-ip)[<i>if-name</i>]# ipaddress label {cellmodem dhcp}	Creates IP address label. Make sure to select the correct IP address type for the modem, as specified in Table 17 on page 206.
4	node(if-ip)[<i>if-name</i>]# port virtual cell-modem port	Enters virtual cellular modem <i>port</i> 's configuration mode.
5	node(vpirt-cell)[port]# bind interface ctx-name if-name	Binds the virtual cellular modem port to the IP interface created above.

Enabling the Cellular Modem

To enable the modem, use the following command:

Mode: port virtual cell-modem

Step	Command	Purpose
1	node(vpirt-cell)[port]# no shutdown	Enables connection to and transmission over the cellular network.

Displaying the Cellular Modem's Configuration and Status

To display the modem's configuration and status, use the following command:

Mode: Operator exec

Step	Command	Purpose
1	node> show port virtual cell-modem port	Shows virtual cellular modem port.

Example 1: Show virtual port with physical modem unplugged

```
node>show port virtual cell-modem 1
port cell-modem1
=====
MDN                :
Vendor ID          : 0x1199
Product ID        : 0x9051
Priority           : 1
Delete On Disconnect : false
APN               :
Administrative State : Up
Bound IP Interface : ROUTER.AIRCARD_340U
IP Addresses       : AIRCARD_340U (DHCP): down
                   Requested: (none)
                   Operational: (none)

Operational State  : Down
Hardware Address   :
MTU               : 1492
ARP               : enabled
Multicast         : enabled
Rx Statistics      : 0 bytes in 0 packets
                   0 errors 0 drops, 0 overruns
                   0 multicast packets

Tx Statistics      : 0 bytes in 0 packets
                   0 errors 0 drops, 0 collisions 0 carrier errors
```

Example 2: Show virtual port with physical modem connected to network

```
node>show port virtual cell-modem 1
port cell-modem1
=====

MDN:
Vendor ID:                0x1199
Product ID:               0x9051
Priority:                  1
Delete On Disconnect:     false
APN:
Administrative State:     Up
Bound IP Interface:       ROUTER.AIRCARD_340U
IP Addresses:             AIRCARD_340U (DHCP): up
                           Requested: (none)
                           Operational: 10.4.152.60/30

Operational State:       Up
Hardware Address:        a2:93:38:95:34:c2
Configured MTU:          1492
```

```

Actual MTU:                1500
ARP:                       enabled
Multicast:                 enabled
Rx Statistics:             857 bytes in 7 packets
                           0 errors 0 drops, 0 overruns
                           0 multicast packets
Tx Statistics:             1134 bytes in 11 packets
                           0 errors 0 drops, 0 collisions 0 carrier errors

```

Debugging the Cellular Modem

To see runtime logging related to the cellular modem, use the following commands:

Mode: Operator exec

Step	Command	Purpose
1	<code>node>trace phycellular {level debug [{detail detail full-detail}]}</code>	Shows log messages related to the physical modem.
2	<code>node>trace virtcellular {level debug [{detail detail full-detail}]}</code>	Shows log messages related to the virtual port.
3	<code>node>trace virtual {level debug [{detail detail full-detail}]}</code>	Shows log messages related to attaching a physical modem to a virtual port.

Example 1: Configure a Netgear AirCard 340U

```

node(cfg)#context ip ROUTER
node(ctx-ip)[ROUTER]#interface WAN
node(if-ip)[WAN]#ipaddress WAN dhcp
node(if-ip)[WAN]#no accept hostname
node(if-ip)[WAN]#
node(if-ip)[WAN]#port virtual cell-modem 1
node(vpirt-cell)[1]#link usb vendor 0x1199 product 0x9051 priority 1
node(vpirt-cell)[1]#bind interface ROUTER WAN
node(vpirt-cell)[1]#no shutdown

```

Example 2: Configure a Netgear AirCard 341U

```

node(cfg)#context ip ROUTER
node(ctx-ip)[ROUTER]#interface WAN
node(if-ip)[WAN]#ipaddress WAN cellmodem
node(if-ip)[WAN]#no accept hostname
node(if-ip)[WAN]#
node(if-ip)[WAN]#port virtual cell-modem 1
node(vpirt-cell)[1]#link usb vendor 0x1199 product 0x9055 priority 1
node(vpirt-cell)[1]#bind interface ROUTER WAN
node(vpirt-cell)[1]#no shutdown

```

Troubleshooting

My cellular modem is not getting an IP address

If your modem is bound to an IP address, and the modem is connected to the network, but the IP address never reaches the up state, make sure that the IP interface's address is the correct type for the modem, accord-

ing to [table 17](#) on page 206.

An example of this issue is shown below:

```
node(cfg)#show port virtual cell-modem 1
port cell-modem1
=====

MDN:
Vendor ID:                0x1199
Product ID:               0x9051
Priority:                  1
Delete On Disconnect:    false
APN:
Administrative State:     Up
Bound IP Interface:       ROUTER.AIRCARD_340U
IP Addresses:             AIRCARD_340U (CELLMODEM): released
                          Requested: (none), ignoring hostname
                          Operational: (none)

Operational State:       Up
Hardware Address:        a2:93:38:95:34:c2
Configured MTU:          1492
Actual MTU:              1500
ARP:                     enabled
Multicast:               enabled
Rx Statistics:           1471 bytes in 9 packets
                          0 errors 0 drops, 0 overruns
                          0 multicast packets
Tx Statistics:           2036 bytes in 14 packets
                          0 errors 0 drops, 0 collisions 0 carrier errors
```

The vendor and product IDs indicate that this virtual port is connected to a Netgear AirCard 340U, which requires an IP address type of DHCP, but the IP address type is configured as CELLMODEM. To fix this, execute the following commands:

```
node(cfg)#context ip ROUTER
node(ctx-ip)[ROUTER]#interface AIRCARD_340U
node(if-ip)[AIRCARD~]#ipaddress AIRCARD_340U dhcp
node(if-ip)[AIRCARD~]#port virtual cell-modem 1
node(vpirt-cell)[1]#shutdown
node(vpirt-cell)[1]#no shutdown
```

My cellular modem has a static public IP address but I can't access my device using it

If you have requested a static public IP address from your cellular carrier, but you are not able to access your device using that IP, you may need to modify your modem's configuration. Trinity does not provide this functionality. You will have to do this from your PC.

For example, if you are using the Netgear AirCard 341U and want to access your Trinity Device using its public IP address, you will need to put it in IP Passthrough Mode. To do this:

1. Plug the modem into your PC.
2. Open a browser and go to <http://192.168.1.1> to open the modem's web interface.
3. Login. The default password is *password*.
4. Click on the *Router* tab (see [figure 23](#)).

5. Click the radio button to enable *IP Passthrough Mode*. The modem will inform you that it must reboot to apply the configuration and prompt you to confirm. Confirm and wait for the modem to reboot.
6. Now you can plug the modem into your Trinity device and the static public IP address will be assigned directly to your Trinity device and it will be accessible from that IP. As with any device connected directly to the Internet, make sure to secure it, e.g. using ACL rules.

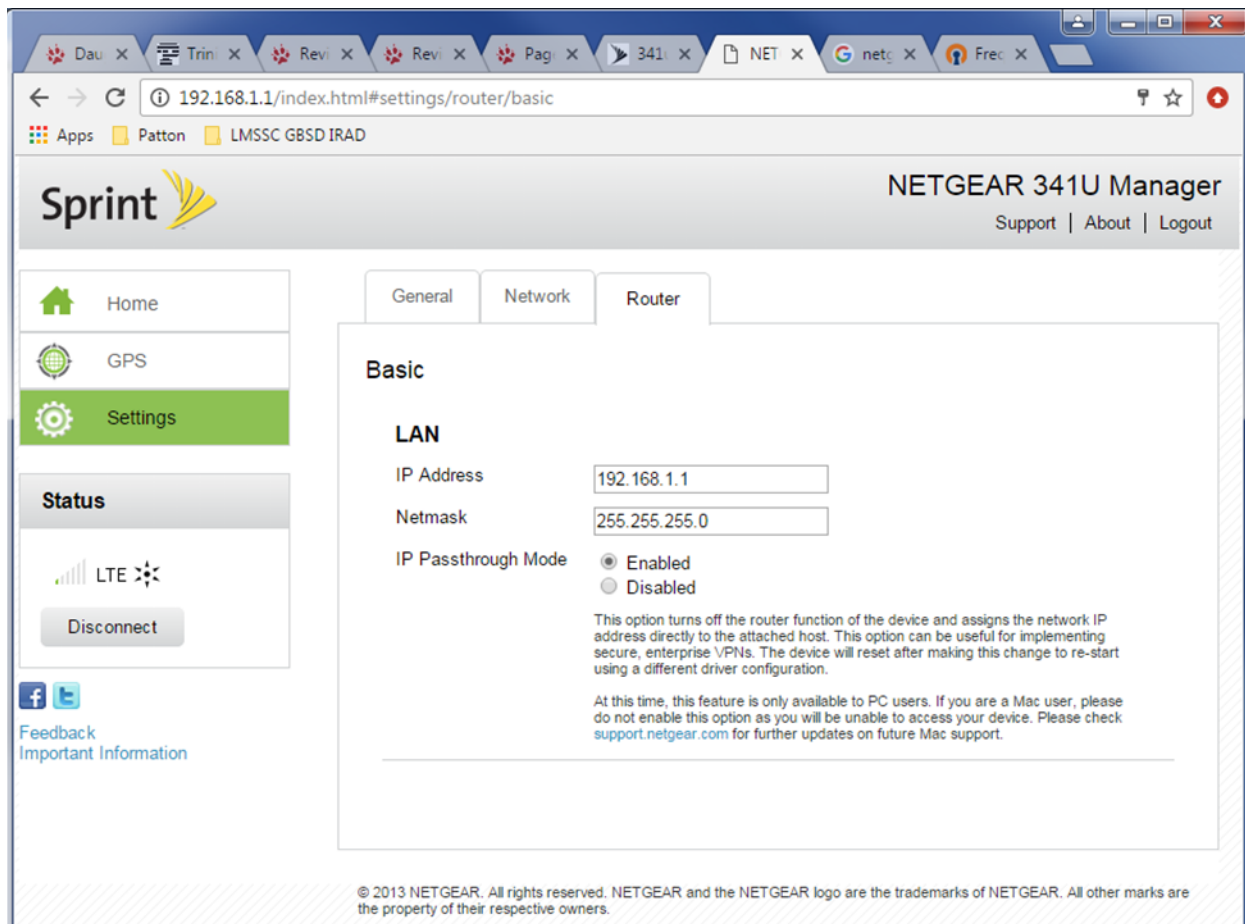


Figure 23. NETGEAR 341U Manager screen

Chapter 16 Hardware Switching

Chapter contents

Introduction	215
Switch Groups	215
Switch Group Configuration Task List	216
Create a switch group	216
Bind switch group to an IP interface	216
Create switch group interfaces	216
Bind ports to switch group interface	217
Examples	217
LAN/WAN Configuration	217
Configuring Two LANs	218
VLAN (802.1p/Q)	218
VLAN configuration task list	219
Configure switch mode	219
Enter switch group interface configuration mode	219
Permit untagged packets	219
Permit tagged packets	220
Encapsulate untagged traffic	220
Encapsulate all traffic	221
Examples	221
Example 1: Interface Isolation	221
Example 2: VLAN Tagging	222
Example 3: Q-in-Q	224
Access Control List Configuration	224
About Access Control Lists (ACLs)	224
What ACLs Do	224
Why You Should Configure ACLs	225
Features of Access Control Lists	225
Access Control List (ACL) Configuration Task List	226
Mapping the Goals of the ACL	226
Creating an ACL Profile and Entering Configuration Mode	226
Adding and Deleting a Filter Rule to the Current ACL Profile	227
Binding and Unbinding an ACL Profile to a Switch Port	230
Displaying an ACL Profile	230
QoS Traffic Scheduler	231
About QoS	231
Packet walkthrough	231
QoS Traffic Scheduler Configuration Task List	232
Create a class of service profile and assign its traffic class	232
Create an access control list profile and create classifier rules	233

Binding an access control list to the receiving switch port	233
Configure traffic classes' scheduling modes	233
Binding a service policy to the transmitting switch port	234
Configuring transmit rate shaping for a switch port	235
Example	236
ToS Stripping and Prioritization.....	237
About ToS stripping and prioritization	237
ToS Stripping and Prioritization Configuration Task List	237
Create a class of service profile and configure its VLAN priority and/or DSCP	238
Create an access control list profile and create classifier rules	238
Binding an access control list to the receiving switch port	238
Example	239
MAC Filter Configuration.....	239
Ethernet Switch MAC Filter Configuration Task List	239
Creating a MAC Filter Profile and Enter Configuration Mode	240
Adding a Filter Rule to the Current MAC Filter Profile	240
Binding and Unbinding a MAC Filter to an Ethernet Switch Port	240
Displaying a MAC Filter Profile	241
Trunk Configuration.....	242
Ethernet Switch Trunk Configuration Task List	242
Creating a Trunk Profile and Enter Configuration Mode	242
Binding and Unbinding a Trunk Profile to an Ethernet Switch Port	242
Displaying an Ethernet Switch Trunk	243
Debugging an Ethernet Switch Trunk	243
Ethernet Service Policy Configuration	244
About QoS	244
Packet Walkthrough	244
Ethernet Switch Service Policy Configuration Task List	246
Configure Traffic Class for Priority Scheduling	246
Configure Traffic Class for Shared Scheduling	246
Binding a Service Policy to an Ethernet Switch Port	246

Introduction

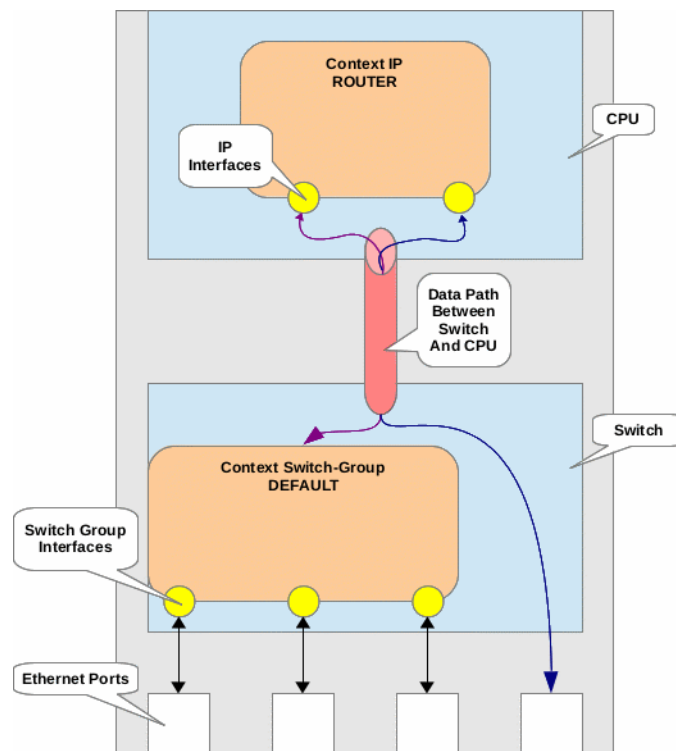
This chapter discusses hardware switching. The following are covered:

- Switch Groups
- VLAN (802.1p/Q)
- Access Control List Configuration
- QoS Traffic Scheduler
- ToS Stripping and Prioritization
- MAC Filter Configuration
- Trunk Configuration
- Ethernet Service Policy Configuration

Switch Groups

This device has an Ethernet switch inside to which external ports are connected. Thus, Ethernet switching between these ports is possible without using the main CPU. This frees the CPU to be used for other tasks, such as IP routing or voice applications.

However, the ports must be configured to use the switch by binding them to switch group interfaces. Otherwise, the switch will forward all packets they receive to the CPU. Consider the diagram below:



- The IP context resides on the CPU indicating that IP routing and termination is performed by the device's main CPU.
- The switch group context resides on the switch, indicating that Ethernet switching is performed by the switch without using CPU resources.
- The first three Ethernet ports are bound to interfaces in the switch group context, indicating that Ethernet traffic is switched between the first three ports, but not the fourth.
- The switch group context is bound to an IP interface, indicating that traffic received by the first three Ethernet ports can be routed out the fourth Ethernet interface. Also, a PC attached to one of the first three Ethernet ports can be used to manage the device.
- Even though the fourth Ethernet port is not bound to a switch group interface, its traffic still goes through the switch before reaching the CPU.
- All ports attached to the switch share a single data path to the CPU. This means that even though traffic can be switched between ports at line rate, routing between ports connected to the switch is limited by the speed of the data path between the switch and CPU.

This chapter describes the tasks involved in configuring a switch group to perform Ethernet switching between the device's ports.

Switch Group Configuration Task List

To configure a switch group, perform the tasks described in the following sections.

Create a switch group

By default, this device has one switch group, `DEFAULT`. Unless you need more than one switch group, use `DEFAULT` rather than create another.

Mode: Configure

Step	Command	Purpose
1	device(cfg)#context switch-group <i>ctx-name</i>	Create switch group <i>ctx-name</i> if it doesn't yet exist and enter "context switch group" mode.

Bind switch group to an IP interface

This step is optional. It is only needed if you want this device to serve as the gateway for this network or if you want this device to be managed from this network.

Mode: Context switch group

Step	Command	Purpose
1	device(ctx-swgrp)[<i>ctx-name</i>]#bind interface [<i>ip-ctx-name</i>] <i>ip-if-name</i>	Bind this switch group to IP interface <i>ip-if-name</i> in IP context <i>ip-ctx-name</i> .

Create switch group interfaces

Perform these steps for each of the ports you want in this switch group.

Mode: Context switch group

Step	Command	Purpose
1	device(ctx-swgrp)[ctx-name]#interface if-name	Create interface <i>if-name</i> in switch group <i>ctx-name</i> if it doesn't yet exist and enter "interface switch group configuration" mode.

Bind ports to switch group interface

Perform these steps for each of the ports you want in this switch group.

Mode: Configure

Step	Command	Purpose
1	device(cfg)#port type slot port	Enter port configuration mode
2	device(prt-type)[slot/port]#bind switch-group ctx-name if-name	Bind this port to interface <i>if-name</i> in switch group <i>ctx-name</i> . It will switch traffic with all other ports bound to interfaces in switch group <i>ctx-name</i> .

Examples

LAN/WAN Configuration

The following example shows how to configure `port ethernet 0 0` as a WAN port and the remaining Ethernet ports as LAN ports. The switch will process all traffic from one device on the LAN destined to another device on the LAN. The CPU will only process traffic that must be routed between the LAN and the WAN.

Notice that `port ethernet 0 0` binds directly to an IP interface, whereas the other Ethernet ports bind to switch group interfaces. The switch group then binds to an IP interface.

```

device(cfg)#context ip ROUTER
device(ctx-ip)#interface LAN
device(if-ip)[LAN]#ipaddress 192.168.1.1/24
device(if-ip)[LAN]#exit
device(ctx-ip)#interface WAN
device(if-ip)[WAN]#ipaddress 10.0.0.1/24
device(if-ip)[WAN]#exit

device(cfg)#context switch-group DEFAULT
device(ctx-swgrp)[DEFAULT]#bind interface ROUTER LAN
device(ctx-swgrp)[DEFAULT]#interface 0_1
device(if-swgrp)[DEFAULT.0_1]#exit
device(ctx-swgrp)[DEFAULT]#interface 0_2
device(if-swgrp)[DEFAULT.0_2]#exit
device(ctx-swgrp)[DEFAULT]#interface 0_3
device(if-swgrp)[DEFAULT.0_3]#exit
device(ctx-swgrp)[DEFAULT]#exit

device(cfg)#port ethernet 0 0
device(prt-eth)[0/0]#bind interface ROUTER WAN
device(prt-eth)[0/0]#exit

device(cfg)#port ethernet 0 1

```

```

device(prt-eth)[0/1]#bind switch-group DEFAULT 0_1
device(prt-eth)[0/1]#exit

device(cfg)#port ethernet 0 2
device(prt-eth)[0/2]#bind switch-group DEFAULT 0_2
device(prt-eth)[0/2]#exit

device(cfg)#port ethernet 0 3
device(prt-eth)[0/3]#bind switch-group DEFAULT 0_3
device(prt-eth)[0/3]#exit

```

Configuring Two LANs

The following example shows how to configure `port ethernet 0 0` and `port ethernet 0 1` as one LAN, and `port ethernet 0 2` and `port ethernet 0 3` as a second LAN. Devices attached to the one LAN will have no access to devices attached to the other LAN.

```

device(cfg)#context switch-group LAN1
device(ctx-swgrp)[LAN1]#interface 0_0
device(if-swgrp)[LAN1.0_0]#exit
device(ctx-swgrp)[LAN1]#interface 0_1
device(if-swgrp)[LAN1.0_1]#exit
device(ctx-swgrp)[LAN1]#exit

device(cfg)#context switch-group LAN2
device(ctx-swgrp)[LAN2]#interface 0_2
device(if-swgrp)[LAN2.0_2]#exit
device(ctx-swgrp)[LAN2]#interface 0_3
device(if-swgrp)[LAN2.0_3]#exit
device(ctx-swgrp)[LAN2]#exit

device(cfg)#port ethernet 0 0
device(prt-eth)[0/0]#bind switch-group LAN1 0_0
device(prt-eth)[0/0]#exit

device(cfg)#port ethernet 0 1
device(prt-eth)[0/1]#bind switch-group LAN1 0_1
device(prt-eth)[0/1]#exit

device(cfg)#port ethernet 0 2
device(prt-eth)[0/2]#bind switch-group LAN2 0_2
device(prt-eth)[0/2]#exit

device(cfg)#port ethernet 0 3
device(prt-eth)[0/3]#bind switch-group LAN2 0_3
device(prt-eth)[0/3]#exit

```

VLAN (802.1p/Q)

This section describes the tasks involved in configuring VLANs on switch group interfaces through the CLI.

VLANs may be used to:

- Isolate interfaces from one another. For example, if devices connected to interfaces ETHERNET_0_0 and ETHERNET_0_1 are in a separate network than devices connected to interfaces ETHERNET_0_2 and

ETHERNET_0_3, VLANs can be used to prevent traffic from being forwarded from one network to the other.

- Indicate to an external device which network a given packet came from by using VLAN tags.

VLAN configuration task list

To configure VLANs on the switch, perform the tasks described in the following sections.

Configure switch mode

In order to configure VLANs on the switching hardware, it must be configured for VLAN mode, as opposed to switch-group mode.

When the switch is put into VLAN mode, it automatically:

- Binds all ports to switch-group DEFAULT, if they are not already bound to it.
- Configures all ports to permit untagged traffic only.

Step	Command	Purpose
1	node(cfg)#switch mode vlans	Put the switch into a mode such that VLANs can be configured on it.

Note Some older hardware did not have the switch that supports VLANs installed. If that is the case, the switch mode command will not be available. You will need to be in VLAN mode instead of Switch Group mode. Issue the command “switch mode vlans” from the configure mode.

Enter switch group interface configuration mode

VLAN configuration takes place in the switch group interface configuration mode. Perform the following steps to enter the switch group interface configuration mode.

Mode: Configure

Step	Command	Purpose
1	node(cfg)#context switch-group <i>if-name</i>	Enter configuration mode for switch group <i>ctx-name</i> .
2	node(ctx-swgrp)[<i>ctx-name</i>]#interface <i>if-name</i>	Enter configuration mode for interface <i>if-name</i> in switch group <i>ctx-name</i> .

Permit untagged packets

To permit untagged packets and to forward them transparently, i.e. without encapsulating them in a VLAN tag, use the **permit untagged** command. This is the default, so you will only need to use this command to return the interface to its default configuration. That is, this command overrides the **permit untagged encapsulate vlan *vlan*** and **permit all encapsulate vlan *vlan*** commands that are listed below.

Mode: Switch group interface configuration

Step	Command	Purpose
1	node(if-swgrp)[ctx-name.if-name]#permit untagged	Permit untagged packets and forward them transparently.

To deny untagged traffic, use the **deny untagged** command. Note that to use this command; you must have previously used the **permit vlan vlan** command to permit tagged traffic. Otherwise, the interface would deny all traffic.

Mode: Switch group interface configuration

Step	Command	Purpose
1	node(if-swgrp)[ctx-name.if-name]#deny untagged	Deny untagged traffic.

Permit tagged packets

To permit tagged packets and to forward them transparently, i.e. without encapsulating them in a second VLAN tag, use the **permit vlan vlan** command. You may enter this command multiple times to add to the set of permitted VLAN IDs. This command overrides the **permit all encapsulate vlan vlan** command that is listed below.

Mode: Switch group interface configuration

Step	Command	Purpose
1	node(if-swgrp)[ctx-name.if-name]#permit vlan vlan	Permit packets tagged with any VLAN ID specified in <i>vlan</i> . <i>vlan</i> may be a single VLAN ID, a list, or a range. Valid VLAN IDs are 0 to 4095, inclusive. Examples are: 100, 100,200, and 100,200..299.

To remove from the set of permitted VLAN IDs, use the **deny vlan vlan** command.

Mode: Switch group interface configuration

Step	Command	Purpose
1	node(if-swgrp)[ctx-name.if-name]#deny vlan vlan	Deny traffic tagged with any VLAN ID specified in <i>vlan</i> . <i>vlan</i> may be a single VLAN ID, a list, or a range. Valid VLAN IDs are 0 to 4095, inclusive. Examples are: 100, 100,200, and 100,200..299.

Encapsulate untagged traffic

To permit untagged packets and to encapsulate them in a VLAN tag, use the **permit untagged encapsulate vlan vlan** command. This command overrides the **permit untagged** command that is listed above and the **permit all encapsulate vlan vlan** command that is listed below.

Mode: Switch group interface configuration

Step	Command	Purpose
1	node(if-swgrp)[ctx-name.if-name]#permit untagged encapsulate vlan vlan	Permit untagged packets and encapsulate them in a VLAN tag with ID <i>vlan</i> .

Encapsulate all traffic

To permit all packets, tagged and untagged, and to encapsulate them in a (second) VLAN tag, use the **permit untagged encapsulate vlan vlan** command. This command overrides the **permit untagged**, **permit vlan vlan**, and **permit untagged encapsulate vlan vlan** commands that are listed above.

Note Not all products support this feature.

Mode: Switch group interface configuration

Step	Command	Purpose
1	node(if-swgrp)[ctx-name.if-name]#permit all encapsulate vlan vlan	Permit all packets and encapsulate them in a (second) VLAN tag with ID <i>vlan</i> .

Examples

Example 1: Interface Isolation

In the following example, interfaces ETHERNET_0_0 and ETHERNET_0_1 are isolated from ETHERNET_0_2 and ETHERNET_0_3. That is, ETHERNET_0_0 and ETHERNET_0_1 form one virtual network and ETHERNET_0_2 and ETHERNET_0_3 for another virtual network. Neither network has access to the other.

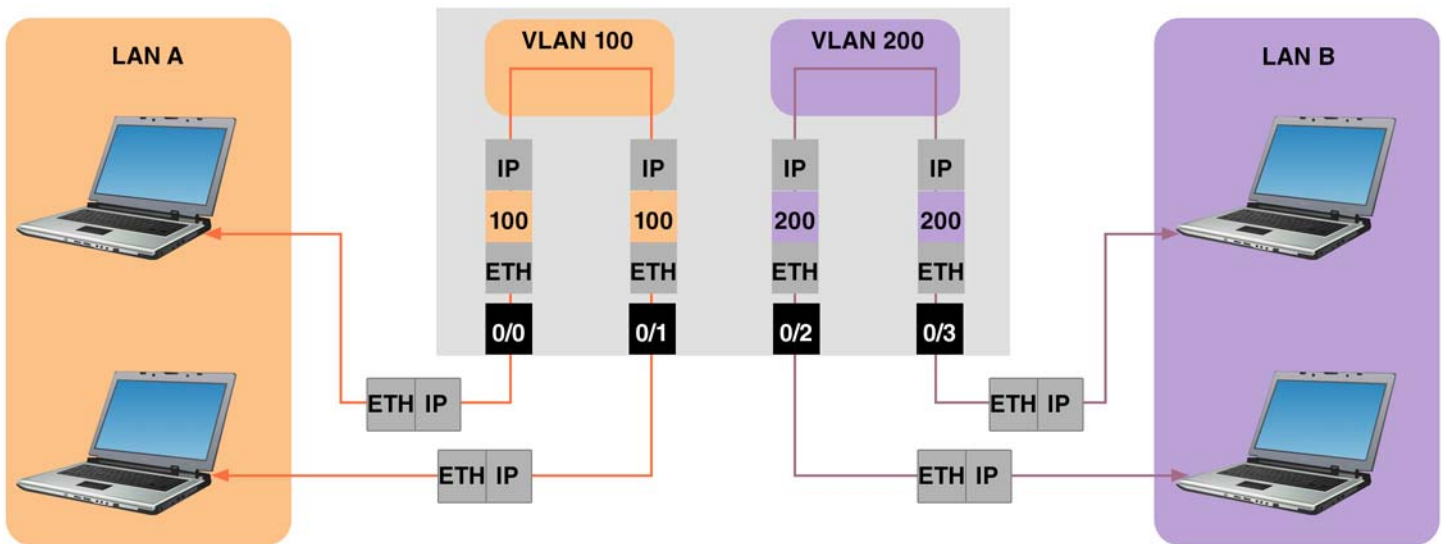


Figure 24. VLAN example 1

```

node(cfg)#context switch-group DEFAULT
node(ctx-swgrp)[DEFAULT]#interface ETHERNET_0_0
node(if-swgrp)[DEFAULT.ETHERNET_0_0]#permit untagged encapsulate vlan 100
node(if-swgrp)[DEFAULT.ETHERNET_0_0]#exit
node(ctx-swgrp)[DEFAULT]#interface ETHERNET_0_1
node(if-swgrp)[DEFAULT.ETHERNET_0_1]#permit untagged encapsulate vlan 100
node(if-swgrp)[DEFAULT.ETHERNET_0_2]#exit
node(ctx-swgrp)[DEFAULT]#interface ETHERNET_0_2
node(if-swgrp)[DEFAULT.ETHERNET_0_2]#permit untagged encapsulate vlan 200
node(if-swgrp)[DEFAULT.ETHERNET_0_2]#exit
node(ctx-swgrp)[DEFAULT]#interface ETHERNET_0_3
node(if-swgrp)[DEFAULT.ETHERNET_0_3]#permit untagged encapsulate vlan 200
node(if-swgrp)[DEFAULT.ETHERNET_0_3]#exit

```

Example 2: VLAN Tagging

In the following example, DSL_0_0, DSL_0_1, DSL_0_2, and DSL_0_3 all provide network access to different customers. ETHERNET_0_0 VLAN tags its traffic to indicate to a router which customer the traffic came from.

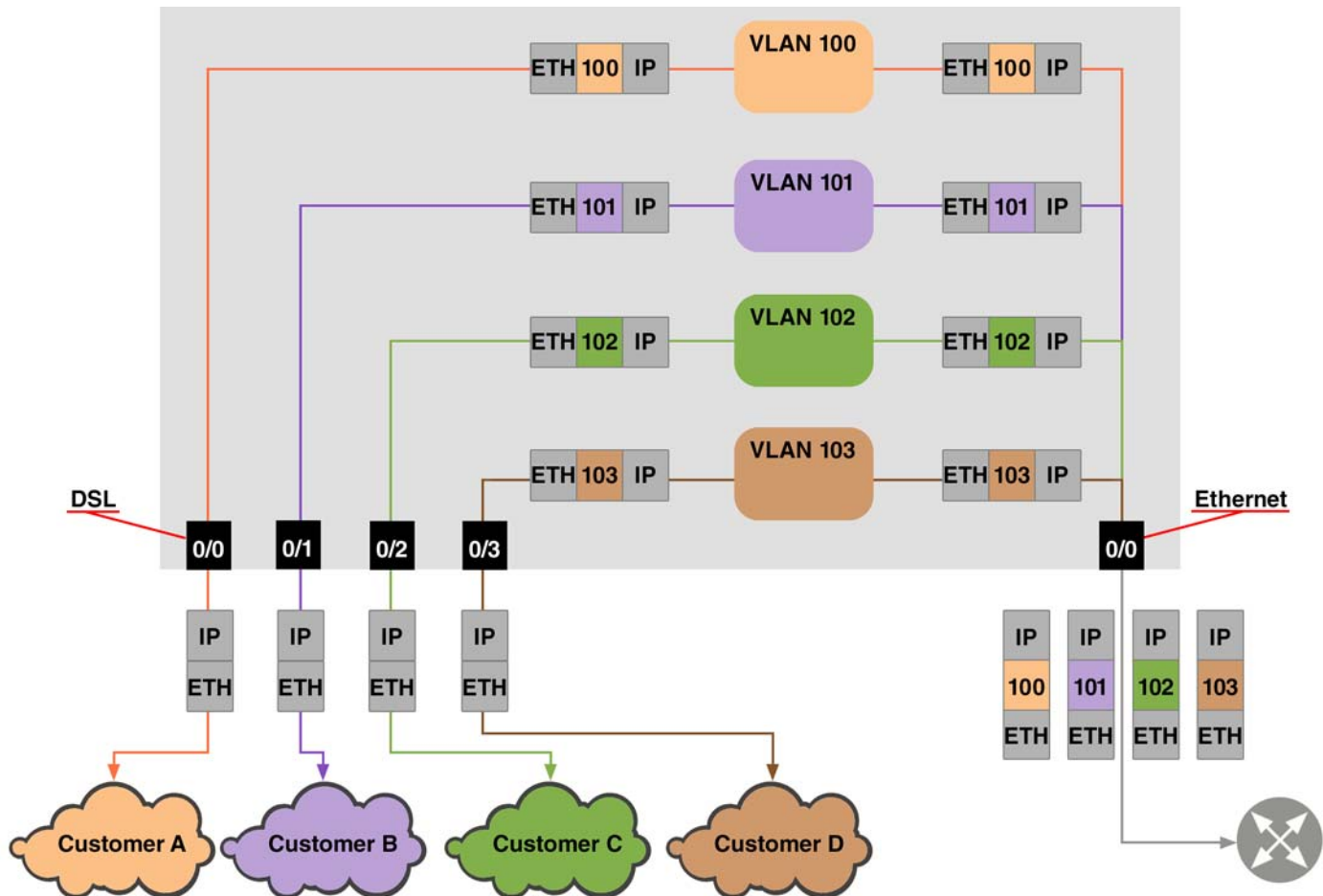


Figure 25. VLAN example 2

```

node(cfg)#context switch-group DEFAULT
node(ctx-swgrp)[DEFAULT]#interface DSL_0_0
node(if-swgrp)[DEFAULT.DSL_0_0]#permit untagged encapsulate vlan 100
node(if-swgrp)[DEFAULT.DSL_0_0]#exit
node(ctx-swgrp)[DEFAULT]#interface DSL_0_1
node(if-swgrp)[DEFAULT.DSL_0_1]#permit untagged encapsulate vlan 101
node(if-swgrp)[DEFAULT.DSL_0_1]#exit
node(ctx-swgrp)[DEFAULT]#interface DSL_0_2
node(if-swgrp)[DEFAULT.DSL_0_2]#permit untagged encapsulate vlan 102
node(if-swgrp)[DEFAULT.DSL_0_2]#exit
node(ctx-swgrp)[DEFAULT]#interface DSL_0_3
node(if-swgrp)[DEFAULT.DSL_0_3]#permit untagged encapsulate vlan 103
node(if-swgrp)[DEFAULT.DSL_0_3]#exit
node(ctx-swgrp)[DEFAULT]#interface ETHERNET_0_0
node(if-swgrp)[DEFAULT.ETHERNET_0_0]#permit vlan 100..103
node(if-swgrp)[DEFAULT.ETHERNET_0_0]#exit

```

Example 3: Q-in-Q

In the following example, DSL_0_0 and DSL_0_1 provide network access to two different customers. Both customers' networks have their own VLAN schemes which must be preserved, so ETHERNET_0_0 encapsulates traffic in a second VLAN tag so it can keep the customer's VLAN tag, yet still distinguish which customer the traffic came from.

```
node(cfg)#context switch-group DEFAULT
node(ctx-swgrp)[DEFAULT]#interface DSL_0_0
node(if-swgrp)[DEFAULT.DSL_0_0]#permit all encapsulate vlan 100
node(if-swgrp)[DEFAULT.DSL_0_0]#exit
node(ctx-swgrp)[DEFAULT]#interface DSL_0_1
node(if-swgrp)[DEFAULT.DSL_0_1]#permit all encapsulate vlan 200
node(if-swgrp)[DEFAULT.DSL_0_1]#exit
node(ctx-swgrp)[DEFAULT]#interface ETHERNET_0_0
node(if-swgrp)[DEFAULT.ETHERNET_0_0]#permit vlan 100,200
node(if-swgrp)[DEFAULT.ETHERNET_0_0]#exit
```

Access Control List Configuration

This chapter provides an overview of hardware switch Access Control Lists (ACLs) and describes the tasks involved in configuring them.

The hardware switch ACLs covered in this chapter are in contrast to the IPv4/v6 Services ACLs available on many of our router products. Hardware switch ACLs perform the packet filtering on the device's hardware switch, whereas IPv4/v6 Services ACLs perform the packet filtering on the device's main CPU. This results in the following differences:

- Hardware switch ACLs have no CPU overhead, whereas IPv4/v6 Services ACLs do.
- Hardware switch ACLs are stateless, whereas IPv4/v6 Services ACLs are stateful. That is, hardware switch ACLs do not perform connection tracking, so they cannot dynamically permit a packet based on the state of the connection it is part of.

When the term ACL is used in this chapter, it refers to a hardware switch ACL, as opposed to an IPv4/v6 Services ACL.

This chapter includes the following sections:

- About ACLs
- ACL configuration task list (see page 226)

About Access Control Lists (ACLs)

This section briefly describes what ACLs do, why and when you should configure them, and their features.

What ACLs Do

ACLs implement a firewall by filtering network traffic, forwarding or dropping each packet based on the ACL rules and bindings. ACL rules specify the criteria used to match packets, e.g. source/destination MAC address, IP address, and/or TCP ports. ACL bindings determine which ingress interface will be matched.

Note Sophisticated users can sometimes successfully evade or fool basic access lists because no authentication is required.

Why You Should Configure ACLs

You should use ACLs to provide a basic level of security for accessing your network. If you do not configure ACLs on your device, all packets passing through the device could be allowed onto all parts of your network. For example, ACLs can allow one host to access a part of your network, and prevent another host from accessing the same area. In [figure 26](#), host A is allowed to access the Human Resources network and host B is prevented from accessing the Human Resources network.

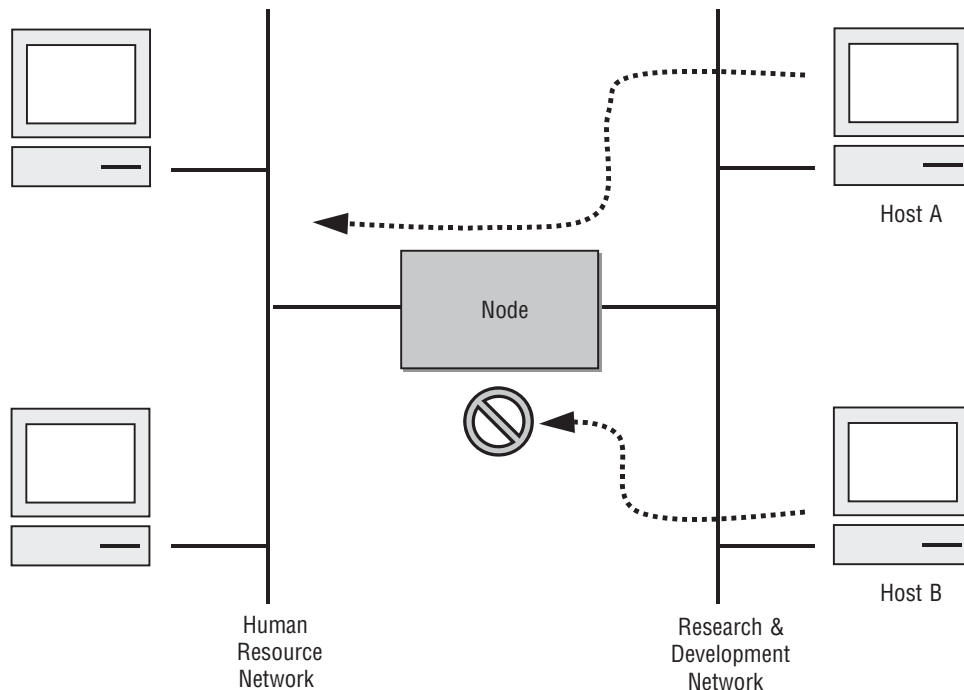


Figure 26. Using traffic filters to prevent traffic from being routed to a network

You can also use ACLs to decide which types of traffic are forwarded or blocked at the device interfaces. For example, you can permit e-mail traffic to be forwarded but at the same time block all Telnet traffic.

Features of Access Control Lists

The following features apply to all ACLs:

- The system may contain a maximum of 415 ACL rules.
- An ACL may contain multiple rules. The order of rules is significant. Each rule is processed in the order it appears in the configuration file. As soon as a rule matches, the corresponding action is taken and no further processing takes place.
- All ACLs have an implicit **deny** all rule at the end. A packet that does not match the criteria of the first rule is subjected to the criteria of the second rule and so on until the end of the ACL is reached, at which point the packet is dropped.
- An empty ACL is treated as an implicit **deny** all list.

Note Two or more administrators should not simultaneously edit the configuration file. This is especially the case with ACLs. Doing this can have unpredictable results.

Once in ACL configuration mode, each command creates a rule in the ACL. When the ACL is applied, the action performed by each rule is one of the following:

- **permit** statement causes any packet matching the criteria to be accepted.
- **deny** statement causes any packet matching the criteria to be dropped.

Access Control List (ACL) Configuration Task List

To configure an ACL, perform the tasks in the following sections.

- Mapping out the goals of the ACL
- Creating an ACL profile and entering configuration mode (see page 226)
- Adding a filter rule to the current ACL profile (see page 227)
- Binding and unbinding an ACL profile to a switch port (see page 230)
- Displaying an ACL profile (see page 230)

Mapping the Goals of the ACL

To create an ACL, you must:

- Assign a unique name to the ACL
- Define packet-filtering criteria

A single ACL can have multiple filtering criteria statements.

Before you begin to enter the commands that create and configure the ACL, be sure that you are clear about what you want to achieve with the firewall. Consider whether it is better to deny specific accesses and permit all others or to permit specific accesses and deny all others.

Note A single ACL can have multiple rules, but editing those rules online can be tedious. Therefore, we recommend editing complex ACLs offline within a configuration file and downloading the configuration file later via TFTP to your device.

Creating an ACL Profile and Entering Configuration Mode

This procedure describes how to create an ACL profile and enter ACL configuration mode.

Mode: Administrator execution

Step	Command	Purpose
1	<code>node(cfg)#profile switch acl <i>name</i></code>	Creates the ACL profile <i>name</i> and enters the configuration mode for this ACL.

The ACL profile name will be known by *name*. Entering this command puts you into *ACL configuration mode* where you can enter the individual statements that will make up the ACL.

Use the **no** form of this command to delete an ACL profile. You cannot delete an ACL profile if it is currently bound to a switch port.

Example: Create an ACL profile

In the following example the ACL profile named FROM_MODEM is created and the shell of the ACL configuration mode is activated.

```
node>enable
node#configure
node(cfg)#profile switch acl FROM_MODEM
node(pf-switch-acl)[FROM_MODEM]#
```

Adding and Deleting a Filter Rule to the Current ACL Profile

The commands **permit** or **deny** are used to define an ACL rule. This procedure describes how to create an ACL rule.

Mode: ACL Configuration

Step	Command	Purpose
1	<code>node(pf-switch-acl)[name]#{permit deny} [src-mac src-mac] [dest-mac dest-mac] [vlan-id vlan-id] [vlan-prio vlan-prio] [dscp dscp] [protocol {arp ip icmp tcp udp sctp}] [src-ip {src-address src-network/src-mask-size src-network src-mask}] [dest-ip {dest-address dest-network/dest-mask-size dest-network dest-mask}] [icmp-type icmp-type [icmp-code]] [src-port src-port] [dest-port dest-port]</code>	Creates an ACL rule that either permits or denies access according to the match criteria.

The table below explains the syntax:

Keyword	Meaning
permit	Forward any packet matching the match criteria.
deny	Drop any packet matching the match criteria.
src-mac	Matches packets from this MAC address, e.g. 00:a0:ba:01:23:45
dest-mac	Matches packets to this MAC address, e.g. 00:a0:ba:0a:bc:de
vlan-id	Matches packets on this VLAN. Note that this matches the VLAN that the switch internally assigns to the packet. For ports that are untagged members or access port members of a VLAN, the internal VLAN does not match the packet's VLAN ID field. See Chapter 14, “Ethernet Port Configuration” on page 197 for details. Valid values are 0 through 4094.
vlan-prio	Matches packets with this IEEE 802.1p VLAN priority field. Note that this will only match packets that were received with a VLAN tag. Valid values are 0 through 7.
dscp	Matches packets with this DiffServ Code Point. Valid values are 0 through 63, or any of the following: af11 af12 af13 af21 af22 af23 af31 af32 af33 af41 af42 af43 cs1 cs2 cs3 cs4 cs5 cs6 cs7 ef. Requires protocol ip, icmp, tcp, udp, or sctp to be specified.
arp	Matches ARP packets

Keyword	Meaning
ip	Matches IPv4 packets
icmp	Matches ICMP packets
tcp	Matches TCP packets
udp	Matches UDP packets
sctp	Matches SCTP packets
src-address	Matches packets from this IPv4 host, e.g. 192.168.1.1. Requires protocol ip, icmp, tcp, udp, or sctp to be specified.
src-network/src-mask-size	Matches packets from this IPv4 subnet, e.g. 192.168.1.0/24. Requires protocol ip, icmp, tcp, udp, or sctp to be specified.
src-network src-mask	Alternate syntax to match packets from this IPv4 subnet, e.g. 192.168.1.0 255.255.255.0. Requires protocol ip, icmp, tcp, udp, or sctp to be specified.
dest-address	Matches packets to this IPv4 host, e.g. 192.168.2.1. Requires *protocol ip, icmp, tcp, udp, or sctp to be specified.
dest-network/dest-mask-size	Matches packets to this IPv4 subnet, e.g. 192.168.2.0/24. Requires protocol ip, icmp, tcp, udp, or sctp to be specified.
dest-network dest-mask	Alternate syntax to match packets from this IPv4 subnet, e.g. 192.168.2.0 255.255.255.0. Requires protocol ip, icmp, tcp, udp, or sctp to be specified.
icmp-type	Matches packets with this ICMP type. Valid values are 0 through 255, or any of the following: echo-reply destination-unreachable redirect alternate-host-address echo-request router-advertisement router-selection time-exceeded parameter-problem timestamp-request timestamp-reply information-request information-reply address-mask-request address-mask-reply traceroute datagram-conversion-error mobile-host-redirect ipv6-where-are-you ipv6-i-am-here mobile-registration-request mobile-registration-reply skip photuris. Requires protocol ip , to be specified.
icmp-code	Matches packets with this ICMP code. Valid values are 0 through 255. Requires protocol ip to be specified.
src-port	Matches packets from this TCP, UDP, or SCTP port. Requires protocol ip, udp, or sctp to be specified.
dest-port	Matches packets to this TCP, UDP, or SCTP port. Requires protocol tcp, udp or sctp to be specified.

If no match criteria is specified, the rule will match all packets, so if you place the rule **deny** at the top of an ACL profile, no packets will pass regardless of the other rules you defined.

Example: Create ACL rules

Select the ACL profile named FROM_MODEM and create some rules to:

- drop all ICMP echo requests (as used by the ping command),
- but forward all other ICMP traffic,
- forward any TCP traffic to host 193.14.2.10 via port 80,

- forward UDP traffic from host 62.1.2.3 to host 193.14.2.11 via any port in the range from 1024 to 2048, and
- forward all traffic from the 97.123.121.0/24 subnet to host 193.14.2.11.

```
node(cfg)#profile switch acl FROM_MODEM
node(pf-switch-acl)[FROM_MODEM]#deny protocol icmp icmp-type echo-request
node(pf-switch-acl)[FROM_MODEM]#permit protocol icmp
node(pf-switch-acl)[FROM_MODEM]#permit protocol tcp dest-ip 193.14.2.10 dest-port
80
node(pf-switch-acl)[FROM_MODEM]#permit protocol udp src-ip 62.1.2.3 dest-ip
193.14.2.11 dest-port 1024..2048
node(pf-switch-acl)[FROM_MODEM]#permit protocol ip src-ip 97.123.121.0/24 dest-ip
193.14.2.11
node(pf-switch-acl)[FROM_MODEM]#exit
node(cfg)#
```

The **no** form of the **permit** or **deny** command is used to delete an ACL rule. This procedure describes how to delete an ACL rule.

Mode: ACL Configuration

Step	Command	Purpose
1	<code>node(pf-switch-acl)[name]#show profile switch acl name</code>	Show the ACL rules in the ACL <i>name</i> . Use this command to get the index of the rule you want to delete. You will use it in step 2.
2	<code>node(pf-switch-acl)[name]#no permit index</code>	Removes the ACL rule at the specified <i>index</i> .

The table below explains the syntax:

Keyword	Meaning
permit	Delete a permit rule.
deny	Delete a drop rule.
index	The order in the ACL of the rule to delete.

Example: Delete an ACL rule

Select the ACL profile named FROM_MODEM and delete the rule to permit any TCP traffic to host 193.14.2.10 via port 80.

```
node(cfg)#profile switch acl FROM_MODEM
node(pf-switch-acl)[FROM_MODEM]#show profile switch acl FROM_MODEM
deny 1 protocol icmp icmp-type echo-request
permit 2 protocol icmp
permit 3 protocol tcp dest-ip 193.14.2.10 dest-port 80
permit 4 protocol udp src-ip 62.1.2.3 dest-ip 193.14.2.11 dest-port 1024..2048
permit 5 src-ip 97.123.121.0/24 dest-ip 193.14.2.11
node(pf-switch-acl)[FROM_MODEM]#no permit 3
node(pf-switch-acl)[FROM_MODEM]#exit
node(cfg)#
```

Binding and Unbinding an ACL Profile to a Switch Port

The command **use** is used to bind an ACL profile to a switch port. This procedure describes how to bind an ACL profile to incoming packets on a switch port.

Mode: Switch Port

Step	Command	Purpose
1	node(switch-port-type)[slot/port]#use acl name in	Binds ACL profile <i>name</i> to incoming packets on port <i>type slot port</i> .

The **no** form of the **use** command is used to unbind an ACL profile from a port. This procedure describes how to unbind an ACL profile from a port.

Mode: Switch Port

Step	Command	Purpose
1	node(switch-port-type)[slot/port]#no use acl in	Unbinds the ACL profile from incoming packets on port <i>type slot port</i> .

The table below explains the syntax:

Keyword	Meaning
type	The type of the port to which the ACL profile is bound, e.g. ethernet or dsl.
slot	The port number of the port to which the ACL profile is bound.
port	The port number of the port to which the ACL profile is bound.
name	The name of an ACL profile that has already been created using the profile switch acl command. This is not used for the no form of the command.

Example: Bind and unbind an ACL profile to/from a port

Bind ACL profile FROM_MODEM to incoming packets on the port ethernet 0 0.

```
node(cfg)#port ethernet 0 0
node(prt-eth)[0/0]#switch
node(switch-port-ethernet)[0/0]#use acl FROM_MODEM in
```

Unbind the ACL profile from incoming packets on the port ethernet 0 0.

```
node(cfg)#port ethernet 0 0
node(prt-eth)[0/0]#switch
node(switch-port-ethernet)[0/0]#no use acl in
```

Displaying an ACL Profile

The **show profile switch acl** command displays the indicated ACL profile. If no specific profile is selected, then all created ACL profiles are shown.

This procedure describes how to display a certain ACL profile.

Mode: Administrator execution or any other mode, except the operator execution mode

Step	Command	Purpose
1	<code>node#show profile switch acl name</code>	Displays the ACL profile <i>name</i> .

Example: Displaying an ACL entry

The following example shows how to display the ACL profile named FROM_MODEM.

```
node#show profile switch acl FROM_MODEM
deny 1 protocol icmp icmp-type echo-request
permit 2 protocol icmp
permit 3 protocol tcp dest-ip 193.14.2.10 dest-port 80
permit 4 protocol udp src-ip 62.1.2.3 dest-ip 193.14.2.11 dest-port 1024..2048
permit 5 src-ip 97.123.121.0/24 dest-ip 193.14.2.11
```

QoS Traffic Scheduler

Some products include an internal Ethernet switch to which the external ports are connected. This switch provides efficient bridging between the Ethernet and Ethernet-like ports. This chapter explains how to configure the switch to provide a differing quality of service (QoS) to packets of different types.

About QoS

There are two main aspects to implementing QoS: packet classification and packet scheduling.

Packet classification is performed by an access control list. In addition to filtering packets, as described in the Chapter 14, “[Hardware Switching: Access Control List Configuration](#)” on page 102, an access control list can also assign packets to a class of service profile.

The class of service profile then assigns the packet to a traffic class.

Packet scheduling is performed by a service policy. The service policy schedules packets for transmission based on the packet's traffic class.

In addition, you may optionally configure rate shaping on the transmit port. Rate shaping can be used in conjunction with packet scheduling, but it can also be used independently.

Packet walkthrough

A packet received by an Ethernet switch port is processed as follows:

1. If the port is bound to an access control list, that access control list assigns the packet to a class of service profile.
2. The class of service profile assigns the packet to a traffic class.
3. The switch selects a port to transmit the packet out of based on the packet's destination MAC address.
4. The port enqueues the packet for transmission based on the packet's traffic class. The port has 8 transmit queues, one for each transmit class. If the port's transmit queue that corresponds to the packet's traffic class is full, then the packet is dropped. This is called tail-dropping.

5. When the port is finished transmitting a packet, if its transmit rate shaper is enabled, it waits to dequeue the next packet such that the bandwidth will remain under the specified rate. Otherwise it proceeds immediately to the next step.
6. The port dequeues a packet from one of its transmit queues and begins transmitting it. Which transmit queue it dequeues from is based on its service policy's configuration. The service policy configures each traffic class for one of two scheduling algorithms:
 - **Priority:** Packets will be dequeued and transmitted from the traffic class's queue until there is a packet in a higher priority traffic classes queue. (Traffic class 7 is highest priority and 0 is lowest.) This means that packets assigned to lower priority traffic classes will never be transmitted as long as there are packets assigned to this traffic class. Lower priority traffic classes queues will fill up and packets assigned to those traffic classes will be dropped if there is enough traffic assigned to the higher priority traffic class.
 - **Shared (shaped deficit weighted round robin, or SDWRR):** Traffic classes that are configured for shared will share the bandwidth available while the higher priority traffic classes' queues are empty. Each shared traffic class is guaranteed at least its configured percentage of the available bandwidth, but may use more if the other traffic classes require less bandwidth than their configured percentage.

Note Traffic classes are configured for shared scheduling should be contiguous or else the behavior will be unexpected. For example, it is acceptable for traffic classes 0 and 6-7 to be configured for priority and traffic classes 1-5 to be configured for shared, but it is not acceptable for classes 1 and 6-7 to be configured for priority and traffic classes 0 and 2-5 to be configured for shared because 0 and 2 are not contiguous.

There are 4 different service policies that may each be configured independently. Each Ethernet switch port is assigned to one of these 4 service policies.

QoS Traffic Scheduler Configuration Task List

To configure a service policy, perform the tasks in the following sections:

- Create a class of service profile and assign its traffic class (see page 232)
- Create an access control list profile and create classifier rules (see page 233)
- Binding an access control list to the receiving switch port (see page 233)
- Configure traffic classes' scheduling modes (see page 233)
- Binding a service policy to the transmitting switch port (see page 234)
- Configuring transmit rate shaping for a switch port (see page 235)

Create a class of service profile and assign its traffic class

This procedure describes how to create a class of service profile and assign its traffic class. The traffic class, together with the service policy used by the egress port, determines what priority will be given to the packet internally within the switch. Higher priority traffic should be assigned to higher traffic classes.

Mode: Configure

Step	Command	Purpose
1	node(cfg)#profile switch cos <i>cos-name</i>	Creates the class of service profile <i>cos-name</i> and enters its configuration mode.
2	node(pf-switch-cos)[cos-name]#traffic-class <i>traffic-class</i>	Configures the traffic class to which packets will be assigned. Valid values are 0 to 7, inclusive, which higher numbers being treated with higher priority.

Example: Creating a class of service profile and assigning its traffic class

Create class of service WEB with traffic class 1, giving it the second to lowest priority.

```
node(cfg)#profile switch cos WEB
node(pf-switch-cos)[WEB]#traffic-class 1
```

Create an access control list profile and create classifier rules

This procedure describes how to create an access control list profile and create rules to classify traffic.

Mode: Configure

Step	Command	Purpose
1	node(cfg)#profile switch acl <i>acl-name</i>	Creates the access control list <i>acl-name</i> and enters its configuration mode
2	node(pf-switch-acl)[acl-name]#permit <i>match set cose cos-name</i>	Assign packets matching <i>match</i> criteria to existing class of service profile <i>cos-name</i> . See “Adding and Deleting a Filter Rule to the Current ACL Profile” on page 106 for details on <code>//match//</code> syntax.

Binding an access control list to the receiving switch port

This procedure describes how to bind an access control list to the switch port that will receive the packets.

Mode: Configure

Step	Command	Purpose
1	node(cfg)#port <i>type slot port</i>	Enters port configuration mode.
2	node(prt-type)[slot /port]#switch	Enters Ethernet switch port configuration mode.
3	node(port-switch-type)[slot /port]#use acl <i>acl-name in</i>	Configure the switch port to classify incoming traffic according to access control list <i>acl-name</i>

Configure traffic classes' scheduling modes

This procedure describes how to configure a traffic classes scheduling modes.

Mode: Configure

Step	Command	Purpose
1	node(cfg)#profile switch service-policy <i>srvpl-name</i>	Enters the configuration mode for service policy <i>srvpl-name</i>
2	node(pf-srvpl)[<i>srvpl-name</i>]#source traffic-class <i>traffic-class</i>	Enters the configuration mode for the traffic class <i>traffic-class</i> within service policy <i>srvpl-name</i>
3	node(src)#[<i>srvpl-name.traffic-class</i>]#{priority share <i>percent</i>}	Configures all ports that use service policy <i>srvpl-name</i> to dequeue from their transmit queue for traffic class <i>traffic-class</i> using either the priority algorithm, or the share algorithm with weight <i>percent</i> .

Example: Configuring traffic classes' scheduling modes

Use priority scheduling for highest priority traffic class (7) and share traffic equally among traffic classes 0-3.

```
node(cfg)#profile switch service-policy 0
node(pf-srvpl)[0]#source traffic-class 7
node(src)[0.7]#priority
node(src)[0.7]#source traffic-class 3
node(src)[0.3]#share 25
node(src)[0.3]#source traffic-class 2
node(src)[0.2]#share 25
node(src)[0.2]#source traffic-class 1
node(src)[0.1]#share 25
node(src)[0.1]#source traffic-class 0
node(src)[0.0]#share 25
```

Binding a service policy to the transmitting switch port

This procedure describes how to bind a service policy to a switch port that will transmit the packets.

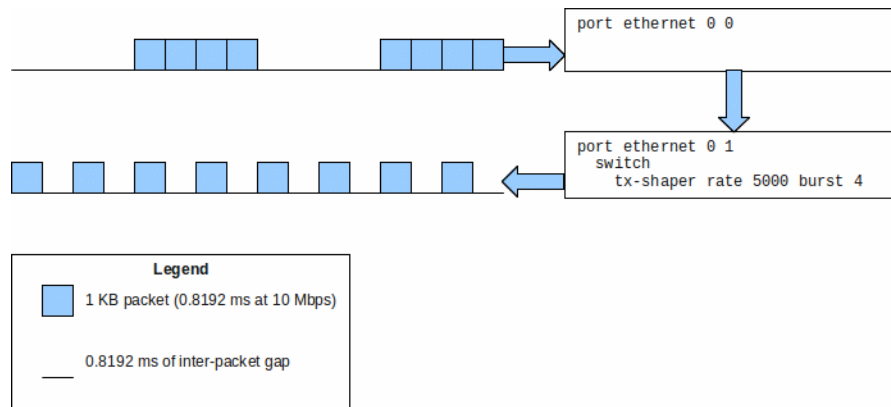
By default all ports use service policy 0.

Mode: Configure

Step	Command	Purpose
1	node(cfg)#port <i>type slot port</i>	Enters port configuration mode.
2	node(prt-<i>type</i>)[<i>slot /port</i>]#switch	Enters Ethernet switch port configuration mode.
3	node(port-switch-<i>type</i>)[<i>slot /port</i>]#use service-policy <i>srvpl-name</i>	Configure the switch port to schedule outgoing traffic according to service policy <i>srvpl-name</i>

Configuring transmit rate shaping for a switch port

This is an optional task where, when configured, the port will limit its transmit bandwidth to the specified rate. It will also absorb the specified burst before it drops packets. The following diagram illustrates how the transmit shaper works:



In the diagram, suppose both Ethernet 0/0 and Ethernet 0/1 are running at 10 Mbps. Ethernet 0/1 is configured to shape its traffic at 5000 kilobits per second (i.e. 5 megabits per second) and to absorb up to 4 kilobytes of burst. A station connected to Ethernet 0/0 is transmitting 1 kilobyte packets in bursts of 4 and then pausing for 4 packet times so that its average rate over the course of a second is 5 megabits per second. Ethernet 0/1 shapes or “smooths” the traffic so that it transmits at a constant 5 megabits per second.

This might be useful if Ethernet 0/1 were connected to a DSL modem that had a 5 megabit per second connection to the far end. If the DSL modem had only a two packet buffer, and if Ethernet 0/1 were to transmit the 4 packet bursts, the DSL modem would receive the first packet in the burst and still be transmitting it when the second packet arrived, so the second packet would be dropped.

While most modern DSL modems would have more than 4 kilobits of packet buffers, it is still possible that the internal Ethernet switch has more buffer capacity than an external device. The internal Ethernet switch allows up to a 16 megabyte burst to be configured, which many external devices would not have.

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg)#port type slot port</code>	Enters port configuration mode.
2	<code>node(prt-type)[slot /port]#switch</code>	Enters Ethernet switch port configuration mode.
3	<code>node(port-switch-type)[slot /port]#tx-shaper rate rate burst burst</code>	Configures transmit shaping on the given port. The rate is specified in kilobits per second, and the burst is specified in kilobytes. The supported rates are based on the burst size, so if an unsupported rate is given the system will automatically select the closest supported rate.

Example: Configuring transmit rate shaping for a switch port

The following example shows how to configure the Ethernet port on slot 0 and port 0 to shape its transmit traffic to 1 megabit per second and to absorb up to 128 kilobytes of burst.

```
node(cfg)#port ethernet 0 0
```

```
node(prt-eth)[0/0]#switch
node(switch-port-ethernet)[0/0]#tx-shaper rate 1000 burst 128
```

Example

The following example provides the following quality of service to packets received by port ethernet 0 0 and transmitted by port ethernet 0 1 depending on their DiffServ code point:

- Any packets arriving with a DSCP of EF (expedited forwarding) will be forwarded immediately.
- Any packets arriving with a DSCP of AFX1 (assured forwarding) will share the bandwidth left over from the EF traffic. AF41 is assured of at least 40% of the remaining bandwidth, AF31 is assured of at least 30%, AF21 is assured of at least 20%, and AF11 is assured of at least 10%.
- Any other traffic is given “best effort,” i.e. any bandwidth remaining after all EF and AF traffic have been transmitted will be given to it.

Note that the access control list profile is bound to the receiving port (in this example, port ethernet 0 0), and the service policy profile is bound to the transmitting port (in this example, port ethernet 0 1). In addition, port ethernet 0 1 is connected to a DSL modem running at 5.7 Mbps, so we enable its rate shaper to limit its transmit rate to 5.7 Mbps.

```
node>enable
node#configure
node(cfg)#profile switch cos EF
node(pf-switch-cos)[EF]#traffic class 7
node(pf-switch-cos)[EF]#exit
node(cfg)#profile switch cos AF41
node(pf-switch-cos)[AF41]#traffic class 4
node(pf-switch-cos)[AF41]#exit
node(cfg)#profile switch cos AF31
node(pf-switch-cos)[AF31]#traffic class 3
node(pf-switch-cos)[AF31]#exit
node(cfg)#profile switch cos AF21
node(pf-switch-cos)[AF21]#traffic class 2
node(pf-switch-cos)[AF21]#exit
node(cfg)#profile switch cos AF11
node(pf-switch-cos)[AF11]#traffic class 1
node(pf-switch-cos)[AF11]#exit
node(cfg)#profile switch cos BE
node(pf-switch-cos)[BE]#traffic class 0
node(pf-switch-cos)[BE]#exit
node(cfg)#profile switch acl CLASSIFIER
node(pf-switch-acl)[CLASSIFIER]#permit dscp ef protocol ip set cos EF
node(pf-switch-acl)[CLASSIFIER]#permit dscp af41 protocol ip set cos AF41
node(pf-switch-acl)[CLASSIFIER]#permit dscp af31 protocol ip set cos AF31
node(pf-switch-acl)[CLASSIFIER]#permit dscp af21 protocol ip set cos AF21
node(pf-switch-acl)[CLASSIFIER]#permit dscp af11 protocol ip set cos AF11
node(pf-switch-acl)[CLASSIFIER]#permit set cos BE
node(pf-switch-acl)[CLASSIFIER]#exit
node(cfg)#profile switch service-policy 1
node(pf-srvpl)[1]#source traffic-class 7
node(src)[1.7]#priority
node(src)[1.7]#exit
node(pf-srvpl)[1]#source traffic-class 4
node(src)[1.4]#share 40
```



```

node(src)[1.4]#exit
node(pf-srvpl)[1]#source traffic-class 3
node(src)[1.3]#share 30
node(src)[1.3]#exit
node(pf-srvpl)[1]#source traffic-class 2
node(src)[1.2]#share 20
node(src)[1.2]#exit
node(pf-srvpl)[1]#source traffic-class 1
node(src)[1.1]#share 10
node(src)[1.1]#exit
node(pf-srvpl)[1]#source traffic-class 0
node(src)[1.0]#priority
node(src)[1.0]#exit
node(pf-srvpl)[1]#exit
node(cfg)#port ethernet 0 0
node(prt-eth)[0/0]#switch
node(switch-port-ethernet[0/0]#use acl CLASSIFIER in
node(cfg)#port ethernet 0 1
node(prt-eth)[0/1]#switch
node(switch-port-ethernet[0/1]#use service-policy 1
node(switch-port-ethernet[0/1]#tx-shaper rate 5700 burst 128

```

ToS Stripping and Prioritization

Some products include an internal Ethernet switch to which the external ports are connected. This switch provides efficient bridging between the Ethernet and Ethernet-like ports. This chapter describes how to configure the switch to set the VLAN priority and/or DSCP fields of transmitted packets to indicate to the receiving device what quality of service (QoS) to provide to each packet. (Note that these fields are only indications, so it depends on the receiving device supporting and being configured for QoS.)

Note Hardware Switching applies to the ForeFront product series only.

About ToS stripping and prioritization

There are two common methods used to indicate to other devices what quality of service should be given to packets:

- VLAN priority field
- IP DSCP field (replaces the older IP ToS field)

Setting either of these fields is referred to as ToS stripping and prioritization. This is implemented as follows:

- Packet classification is performed by an access control list, In addition to filtering packets, as described in the chapter 14, “[Hardware Switching: Access Control List Configuration](#)” on page 102, an access control list can also assign packets to a class of service profile.
- The class of service profile then assigns the packet's VLAN priority and/or DSCP field value(s).

ToS Stripping and Prioritization Configuration Task List

To configure VLAN priority or DSCP packet modification, perform the tasks in the following sections:

- Create a class of service profile and configure its VLAN priority and/or DSCP (see page 238)

- Create an access control list profile and create classifier rules (see page 238)
- Binding an access control list to the receiving switch port (see page 238)

Create a class of service profile and configure its VLAN priority and/or DSCP

This procedure describes how to create a class of service profile and configure its VLAN priority and/or DSCP.

If the VLAN priority is configured, then any packet assigned to this class of service will be transmitted with its VLAN priority field set to the configured value. Otherwise, it will be transmitted with the same VLAN priority field it arrived with. Note that the VLAN priority field is part of the VLAN header, so if the packet is transmitted untagged, this configuration has no effect.

If the DSCP is configured, then any packet assigned to this class of service will be transmitted with its DSCP field set to the configured value. Otherwise, it will be transmitted with the same DSCP field it arrived with. Note that the DSCP field is part of the IP header, so if the packet is not an IP packet, this configuration has no effect.

Mode: Configure

Step	Command	Purpose
1	device(cfg)#profile switch cos <i>cos-name</i>	Creates the class of service profile <i>cos-name</i> and enters its configuration mode.
2	device(pf-switch-cos)[cos-name]#set layer2 cos <i>priority</i>	Optional. Configures the value placed in the VLAN tag's priority field. Valid values are 0 to 7, inclusive.
3	device(pf-switch-cos)[cos-name]#set ip dscp <i>dscp</i>	Optional. Configures the value placed in the IP header's DSCP field. Valid values are 0 to 63, inclusive, or any of the following: af11 af12 af13 af21 af22 af23 af31 af32 af33 af41 af42 af43 cs1 cs2 cs3 cs4 cs5 cs6 cs7 ef

Create an access control list profile and create classifier rules

This procedure describes how to create an access control list profile and create rules to classify traffic.

Mode: Configure

Step	Command	Purpose
1	device(cfg)#profile switch acl <i>acl-name</i>	Creates the ACL profile <i>acl-name</i> and enters the configuration mode for this ACL.
2	device(pf-switch-acl)[acl-name]#permit <i>match set cose cos-name</i>	Assign packets matching <i>match</i> criteria to existing class of service profile <i>cos-name</i> . See “Adding and Deleting a Filter Rule to the Current ACL Profile” on page 106 for details on <i>match</i> syntax.

Binding an access control list to the receiving switch port

This procedure describes how to bind an access control list to the switch port that will receive the packets.

Mode: Configure

Step	Command	Purpose
1	device(cfg)#port <i>type slot port</i>	Enters port configuration mode.
2	device(prt-type)[slot /port]#switch	Enters Ethernet switch port configuration mode.
3	device(port-switch-type)[slot /port]#use acl <i>acl-name in</i>	Configure the switch port to classify incoming traffic according to access control list <i>acl-name</i>

Example

The following example causes packets received by port ethernet 0 0 to be transmitted with VLAN priority and DSCP as follows:

- SSH and Telnet (TCP ports 22 and 23, respectively) are used for device management on this network, so they are transmitted with VLAN priority 3 (critical applications) and DSCP EF (expedited forwarding) to tell other devices to give them higher priority.
- Everything else is considered data, so it is transmitted with VLAN priority 0 (best effort) and DSCP 0 (best effort) to tell other devices to give them lower priority.

```
device>enable
device#configure
device(cfg)#profile switch cos MGMT
device(pf-switch-cos)[VOICE]#set layer2 cos 3
device(pf-switch-cos)[VOICE]#set ip dscp ef
device(pf-switch-cos)[VOICE]#exit
device(cfg)#profile switch cos DATA
device(pf-switch-cos)[VOICE]#set layer2 cos 0
device(pf-switch-cos)[VOICE]#set ip dscp 0
device(pf-switch-cos)[VOICE]#exit
device(cfg)#profile switch acl CLASSIFIER
device(pf-switch-acl)[CLASSIFIER]#permit protocol tcp dest-port 22 set cos MGMT
device(pf-switch-acl)[CLASSIFIER]#permit protocol tcp dest-port 23 set cos MGMT
device(pf-switch-acl)[CLASSIFIER]#permit set cos DATA
device(pf-switch-acl)[CLASSIFIER]#exit
device(cfg)#port ethernet 0 0
device(prt-eth)[0/0]#switch
device(switch-port-ethernet[0/0]#use acl CLASSIFIER in
```

MAC Filter Configuration

Some products include an internal Ethernet switch to which the external ports are connected. This switch provides efficient bridging between the Ethernet and Ethernet-like ports. These ports support blocking received packets based on the source MAC address, destination MAC address, VLAN or any combination thereof. This chapter describes the tasks involved in configuring the switch's ports to filter traffic based on the Ethernet header information.

Note Hardware Switching applies to the ForeFront product series only.

Ethernet Switch MAC Filter Configuration Task List

To configure the Ethernet switch port, perform the tasks described in the following sections:

- Creating a MAC filter profile and enter configuration mode (see page 240)
- Adding a filter rule to the current MAC filter profile (see page 240)
- Binding and unbinding a MAC filter profile to an Ethernet switch port (see page 240)
- Displaying a MAC filter profile (see page 241)

Creating a MAC Filter Profile and Enter Configuration Mode

Mode: Configure

Step	Command	Purpose
1	device(cfg)#profile switch mac-filter <i>name</i>	Creates the MAC filter profile <i>name</i> and enters the configuration mode for this profile. By default, the MAC filter has no rules and will block all received packets.

Adding a Filter Rule to the Current MAC Filter Profile

Mode: Profile MAC filter

Step	Command	Purpose
1	device(pf-switch-mac-filter)#permit { <i>src</i> any } { <i>dest</i> any } [<i>vid</i>]	Creates a rule that permits packets matching all of the supplied command options.

Where the syntax is:

- **src**—The source MAC address to be matched, e.g. 00:a0:ba:01:23:45
- **any**—Indicates that any MAC address is matched.
- **dest**—The destination MAC address to be matched, e.g. 00:a0:ba:01:23:45
- **vid**—The VLAN ID to be matched. Valid values are 2 through 4094. Note that this matches against the VLAN; the switch assigns to the packet for internal processing. For ports that are untagged members or access port members of a VLAN, the internal VLAN does not match the packet's VLAN. See Chapter 14, “Ethernet Port Configuration” for details.

Binding and Unbinding a MAC Filter to an Ethernet Switch Port

The command **use** is used to bind a MAC filter profile to an Ethernet switch port. This procedure describes how to bind a MAC filter profile to incoming packets on an Ethernet switch port.

Mode: Configure

Step	Command	Purpose
1	device(cfg)#port <i>type slot port</i>	Enter port configuration mode.
2	device(prt-type)#[slot /port]#switch	Enter Ethernet switch port configuration mode.
3	device(switch-port-type)#[slot /port]#use mac-filter <i>name</i>	Binds the MAC filter profile <i>name</i> to incoming packets on Ethernet switch port <i>type slot/port</i> .

The **no** form of the **use** command is used to unbind a MAC filter profile from an Ethernet switch port. When using this form, the name of a MAC filter profile represented by the *name* argument above, is not required. This procedure describes how to unbind a MAC filter profile from incoming packets on an Ethernet switch port.

Mode: Ethernet switch port

Step	Command	Purpose
1	device(switch-port-type)#[slot/port]#no use mac-filter	Unbinds the MAC filter profile for incoming packets on Ethernet switch port <i>type slot/port</i> .

Example: Bind and unbind a MAC filter profile to an Ethernet switch port

Create a MAC filter profile that allows only hosts 00:a0:ba:01:23:45 and 00:a0:ba:01:ab:cd to be connected to the Ethernet port on slot 0 and port 0.

```
device(cfg)#profile switch mac-filter FILTER
device(pf-switch-mac-filter)[FILTER]#permit 00:a0:ba:01:23:45 any
device(pf-switch-mac-filter)[FILTER]#permit 00:a0:ba:01:ab:cd any
device(pf-switch-mac-filter)[FILTER]#exit
device(cfg)#port ethernet 0 0
device(prt-eth)[0/0]#switch
device(switch-port-ethernet)[0/0]#use mac-filter FILTER
```

Unbind a MAC filter profile from a port.

```
device(cfg)#port ethernet 0 0
device(prt-eth)[0/0]#switch
device(switch-port-ethernet)[0/0]#no use mac-filter
```

Note When unbinding a MAC filter, the profile *name* argument is not required since only one MAC filter profile can be active at a time on a certain Ethernet switch port.

Displaying a MAC Filter Profile

The **show profile switch mac-filter** command displays the indicated MAC filter profile. If no specific profile is selected then all installed MAC filter profiles are shown. The command shows the profile's rules and the ports that are using the profile.

This procedure describes how to display a certain MAC filter profile.

Mode: Administrator execution or any other mode, except the operator execution mode

Step	Command	Purpose
1	device#show profile switch mac-filter <i>name</i>	Displays the MAC filter profile <i>name</i> .

Example: Displaying a MAC filter profile

The following example shows how to display the MAC filter profile named *FILTER*.

```
device#show profile switch mac-filter FILTER
FILTER
```

```

=====
Source          Destination          VLAN
-----
00:a0:ba:01:23:45
00:a0:ba:01:ab:cd

Port           : Ethernet 0/0

```

Trunk Configuration

Link aggression control protocol (LACP), a.k.a. Trunk, is a protocol used to determine if two or more physical links are connected to the same device and can be used as a single logical link. Some products include an internal Ethernet switch to which the external ports are connected. This switch provides efficient bridging between the Ethernet and Ethernet-like ports. Up to 8 of these ports may be bonded into a single logical link using IEEE 802.1AX link aggregation. This provides both increased bandwidth and redundancy. This is referred to as “trunking.” This chapter describes the tasks involved in configuring the switch's ports for trunking through the CLI.

Note Hardware Switching applies to the ForeFront product series only.

Ethernet Switch Trunk Configuration Task List

To configure an Ethernet switch trunk, perform the tasks described in the following sections:

- Creating a trunk profile and enter configuration mode (see page 242)
- Binding and unbinding a trunk profile to an Ethernet switch port (see page 242)
- Displaying an Ethernet switch trunk (see page 243)
- Debugging an Ethernet switch trunk (see page 243)

Creating a Trunk Profile and Enter Configuration Mode

Mode: Configure

Step	Command	Purpose
1	device(cfg)#profile switch trunk <i>name</i>	Creates the trunk profile <i>name</i> and enter the configuration mode for this profile.

Binding and Unbinding a Trunk Profile to an Ethernet Switch Port

The command **use** is used to bind a trunk profile to an Ethernet switch port.

Mode: Configure

Step	Command	Purpose
1	device(cfg)#port <i>type slot port</i>	Enters port configuration mode.
2	device(prt-<i>type</i>)[<i>slot /port</i>]#switch	Enters Ethernet switch port configuration mode.
3	device(switch-port-<i>type</i>)[<i>slot /port</i>]#use trunk <i>name</i>	Binds trunk profile <i>name</i> to Ethernet switch port <i>type slot/port</i> . All ports that are bound to the same trunk profile are grouped together into a single logical link.

The **no** form of the **use** command is used to unbind a trunk profile from an Ethernet switch port. When using this form, the name of a trunk profile represented by the name argument above, is not required.

This procedure describes how to unbind a trunk profile from an Ethernet switch port.

Mode: Ethernet Switch port

Step	Command	Purpose
1	device(switch-port-type)[slot / port]#no use trunk	Unbinds the trunk profile from the Ethernet switch port <i>type slot/port</i> .

Example: Binding and unbinding a MAC filter profile to an Ethernet switch port

Bond Ethernet 0/1 and Ethernet 0/2 together into a single logical link.

```
device(cfg)#profile switch trunk TRUNK
device(pf-switch-trunk)[FILTER]#exit
device(cfg)#port ethernet 0 1
device(prt-eth)[0/1]#switch
device(switch-port-ethernet)[0/1]#use trunk TRUNK
device(switch-port-ethernet)[0/1]#exit
device(prt-eth)[0/1]#exit
device(cfg)#port ethernet 0 2
device(prt-eth)[0/2]#switch
device(switch-port-ethernet)[0/2]#use trunk TRUNK
```

Displaying an Ethernet Switch Trunk

The **show profile switch trunk** command displays the indicated trunk profile. If no specific profile is selected then all installed trunk profiles are shown. The ports that are using the specified trunk are displayed.

This procedure describes how to display a certain trunk profile.

Mode: Administrator execution or any other mode, except the operator execution mode

Step	Command	Purpose
1	device#show profile switch acl name	Displays the trunk profile <i>name</i> .

Example: Displaying a trunk profile

The following example shows how to display the trunk profile named TRUNK.

```
device#show profile switch trunk TRUNK
TRUNK
=====
Port           : Ethernet 0/1
Port           : Ethernet 0/2
```

Debugging an Ethernet Switch Trunk

The **trace lacp** command is used to debug link aggregation control protocol negotiation during system operation. Use the **no** form of this command to disable any debug output.

This procedure describes how to debug the trunk profiles.

Mode: Administrator execution of any other mode, except the operator execution

Step	Command	Purpose
1	<code>device#trace lacp { debug [detail detail full-detail] level }</code>	Enables the Ethernet switch trunk debug monitor.

Example: Debugging Ethernet switch trunk profiles

The following example enables the debug monitor for Ethernet switch trunk profiles globally.

```
device#trace lacp debug full-detail
```

The following example enables the debug monitor for Ethernet switch trunk profiles globally.

```
device#no trace lacp
```

Ethernet Service Policy Configuration

Some products include an internal Ethernet switch to which the external ports are connected. This switch provides efficient bridging between the Ethernet and Ethernet-like ports such as G.SHDSL and EFM ports. The switch is able to be configured to provide a differing quality of service (QoS) to packets of different types. This chapter provides an overview of Ethernet switch service policies, which are an integral part of QoS configuration.

In addition, see Chapter 38, “[Access Control List Configuration](#)” on page 440 and Chapter 16, “[Hardware Switching](#)” on page 213 for other components that are necessary to configure QoS.

About QoS

There are two main aspects to implementing QoS: packet classification and packet scheduling.

Packet classification is performed by a profile classifier. The classifier is part of the overall Quality-of-Service architecture of Trinity. As depicted in figure 63 on page 424, the classifier groups packet flows into virtual traffic-classes.

Packet Walkthrough

A packet received by an Ethernet switch port is processed as follows:

1. If the port is bound to a profile classifier, that profile classifier assigns the packet to a class of service profile.
2. The class of service profile assigns the packet to a traffic class.
3. The switch selects a port to transmit the packet out of, based on the packets destination MAC address.
4. The port enqueues the packet for transmission based on the packet's traffic class. The port has 8 transmit queues, one for each transmit class. If the port's transmit queue that corresponds to the packet's traffic class is full, then the packet is dropped. This is called tail-dropping.
5. When the port is finished transmitting a packet, it dequeues a packet from one of its transmit queues and begins transmitting it. Which transmit queue it dequeues from is based on its service policy's configuration. The service policy configures each traffic class for one of two scheduling algorithms:
 - **Priority:** Packets will be dequeued and transmitted from the traffic class's queue until there is a packet in a higher priority traffic class's queue. (Traffic class 7 is the highest priority and 0 is the lowest.) This

means that packets assigned to lower priority traffic classes will never be transmitted as long as there are packets assigned to this traffic class. Lower priority traffic classes' queues will fill up and packets assigned to those traffic classes will be dropped if there is enough traffic assigned to the higher priority traffic class.

- **Shared** (shaped deficit weighted round robin): Traffic classes that are configured for shared will share the bandwidth available while the higher priority traffic classes' queues are empty. Each shared traffic class is guaranteed at least its configured percentage of the available bandwidth, but may use more if the other traffic classes require less bandwidth than their configured percentage.

Note Traffic classes configured for shared scheduling should be contiguous or else the behavior will be unexpected. For example, it is acceptable for traffic classes 0 and 6-7 to be configured for priority and traffic classes 1-5 to be configured for shared; however, it is not acceptable for classes 1 and 6-7 to be configured for priority and traffic classes 0 and 2-5 to be configured for shared, because 0 and 2 are not contiguous.

There are 4 different service policies that may each be configured independently. Each Ethernet switch port is assigned to one of these 4 service policies.

Ethernet Switch Service Policy Configuration Task List

To configure a service policy, perform the tasks in the following sections.

Configure Traffic Class for Priority Scheduling

This procedure describes how to configure a traffic class for priority scheduling.

Mode: Configure

Step	Command	Purpose
1	device(cfg)#profile switch service-policy name	Enters the configuration mode for service policy <i>name</i> .
2	device(pf-srvpl) device#source traffic-class traffic-class	Enters the configuration mode for the traffic class <i>traffic-class</i> within service policy <i>name</i> .
3	device(src)[name.traffic-class]#priority	Configures all ports that use service policy <i>name</i> to dequeue from their transmit queue for traffic class <i>traffic-class</i> using the priority algorithm.

Configure Traffic Class for Shared Scheduling

This procedure describes how to configure a traffic class for shared scheduling.

Mode: Configure

Step	Command	Purpose
1	device(cfg)#profile switch service-policy name	Enters the configuration mode for service policy <i>name</i> .
2	device(pf-srvpl) device#source traffic-class traffic-class	Enters the configuration mode for the traffic class <i>traffic-class</i> within service policy <i>name</i> .
3	device(src)[name.traffic-class]#share percent	Configures all ports that use service policy <i>name</i> to dequeue from their transmit queue for traffic class <i>traffic-class</i> using the shaped deficit weighted round robin (SDWRR) algorithm with a weight of <i>percent</i> .

Binding a Service Policy to an Ethernet Switch Port

This procedure describes how to bind a service policy to an Ethernet switch port. By default all ports use service policy 0.

Mode: Configure

Step	Command	Purpose
1	device(cfg)#port <i>type slot port</i>	Enters port configuration mode.
2	device(prt-type)[slot/port]#switch	Enters Ethernet switch port configuration mode.
3	device(port-switch-type)[slot/port]#use service-policy <i>name</i>	Configures the Ethernet switch port to schedule transmission according to service policy <i>name</i> .

Chapter 17 DSL Port Configuration

Chapter contents

Introduction	249
G.SHDSL EFM Setup	249
Configuring the Mode for the G.SHDSL Connection	249
Configuring the Annex Type for the G.SHDSL Connection	250
Configuring the Payload Data Rate for the G.SHDSL Connection	250
Configuring the TCPAM for the G.SHDSL connection	251
DSL Emergency Freeze	251
DSL TX Power Increase	251
DSL Suspect Mode	252
Multiport G.SHDSL Devices	252
Configuring the Profile for the G.SHDSL Connection	252
VDSL/ADSL Configuration.....	253
Introduction	253
Configuration Task List	253
Configure Physical Parameters	254
Configure PTM Protocol Stack	254
IPoE	254
PPPoE	255
Create ATM PVC	256
Configure ATM Protocol Stack	256
IPoEoA	257
PPPoEoA	257
Troubleshooting DSL Connections.....	258
Checking DSL connection status with show command	258
Link State	259
Debugging the DSL connection with the DSL debug command.	259

Introduction

This chapter provides an overview of the DSL ports, their characteristics and the tasks involved in the configuration.

G.SHDSL EFM Setup

Configuring the Mode for the G.SHDSL Connection

An EFM DSL device transports Ethernet packets directly over the connection. A standard DSL connection operates over a 2-wire interface. Where hardware will allow it, DSL wire pairs may be bonded together into one link by changing the service-mode. The available modes are listed below:

- 2-wire – Standard 2-Wire mode. Data is byte-interleaved, where subsequent bytes in the data stream are sent on alternating pairs.
- 4-wire – Enhanced 4-Wire mode, meaning the DSL line pairs can be activated and train independently. If one pair droops from Showtime, the other pair drops and they must both be restarted.
- 8-wire – Enhanced 8-Wire mode, meaning the DSL line pairs can be activated and train independently. If one pair drops from Showtime, the other pair drops and they must both be restarted.

Mode: port dsl 0 0

Step	Command	Purpose
1	device(prt-dsl)[0/0]#service-mode 2-wire 4-wire 8-wire	Define the mode for the DSL connection. Default: 4-wire.

Example: Configuration of the 4-wire service mode

```
port dsl 0 0
service-mode 4-wire
annex-type

bind interface ETH_DSL router
```

On some devices, G.SHDSL ports can be configured as CO or CPE mode.

Mode: port dsl 0 0

Step	Command	Purpose
1	device(prt-dsl)[0/0]#mode co cpe	Define the mode for the DSL connection.

Configuring the Annex Type for the G.SHDSL Connection

In order for the DSL link to connect, Patton devices supporting a 4-wire G.SHDSL card as a CPE client must configure the DSL annex type to match the annex type on the CO side.

Mode: port dsl 0 0

Step	Command	Purpose
1	<code>device(prt-dsl)[0/0]#annex-type a-f b-g</code>	Define the annex type for the DSL connection.

Example: Configuration of the 4-wire annex type

```
port dsl 0 0
service-mode 4-wire
annex-type a-b
bind interface ETH_DSL router
```

Configuring the Payload Data Rate for the G.SHDSL Connection

The payload-rate command allows configuring the bit rate mode to adaptive (commonly used) or specifying a fixed value for the payload data rate. The payload data rate is configured for a single 2-wire pair also in case of a 4-wire connection. The valid range for this command changes based on the “tcpam” command explained later.

Note The payload rate is only valid in 64 Kbps increments as shown in [Table 18](#) on page 251, however the command will allow any integer value in the range to be entered. The entered value will be automatically adjusted to a valid rate.

Mode: port dsl 0 0

Step	Command	Purpose
1	<code>[device](prt-dsl)[0/0]#payload-rate { adaptive [max <192..15296>] <192..15296> }</code>	Define the data rate for the DSL connection. Default: adaptive.

[Table 18](#) on page 251 provides an overview of the available payload rates. When the “payload-rate” command is configured as “adaptive”, the discovered rate can be adjusted with the “snr-margin” command. This adjusts the acceptable SNR of the link.

Step	Command	Purpose
1	<code>[device](prt-dsl)[0/0]#snr-margin <-10..22></code>	Adjust the signal sensitivity of the adaptive payload rate. Default:6

Table 18. Payload Rate Configuration Overview

payload-rate	payload-rate	payload-rate	payload-rate
192	896	1536	2240
256	960	1600	2304
320	1024	1664	2560
384	1088	1728	3392
448	1152	1792	3840
512	1216	1920	5056
576	1280	1984	5696
704	1344	2048	
768	1408	2112	
832	1472	2176	

Configuring the TCPAM for the G.SHDSL connection

The TCPAM setting provides a means to adjust the max configurable data rate. The default setting of “auto(16/32)” allows for the best compatibility with other EFM equipment, and provides a max payload rate of 5696.

Some hardware allows a TCPAM configuration of “auto(64/128)”. This mode is only compatible with some other Patton hardware, but allows for a payload rate of 15296.

Step	Command	Purpose
1	<code>device(prt-dsl)[0/0]#tcpam { auto(16/32) auto(64/128) 4 8 16 32 64 128 }</code>	Configure the TCPAM of the link. Default: auto(16/32)

Note TCPAM-4 and TCPAM-8 provide the DSL ports with longer reach and greater stability in noisy environments. The maximum payload rate is 2496 kbps for TCPAM-4, and 5056 kbps for TCPAM-8. These are non-standard extensions to G.SHDSL, so they are only guaranteed to work with two of these products back-to-back.

DSL Emergency Freeze

This feature allows the LINE to survive "micro disruptions" without retraining the link. The performance of this feature improves with lower data rates, distance, and TCPAM settings.

Mode: port dsl <slot> <port>

Step	Command	Purpose
1	<code>device(prt-dsl)[<slot>/<port>]# [no] emergency-freeze</code>	Configures the emergency-freeze feature to survive micro breaks in the DSL line.

DSL TX Power Increase

This command may be used to to increase the DSL/line ports transmit power up to 0.7 dB. This is non-standard, but it may provide a more stable link in a noisy environment.

Mode: port dsl <slot> <port>

Step	Command	Purpose
1	device(prt-dsl)[0/0]#tx-power-increase <dB>	Increases the ports transmit power dB dBs above what is specified by the standard.
2	device(prt-dsl)[0/0]#no tx-power-increase	Sets the ports transmit power to what is specified by the standard.

DSL Suspect Mode

When one of the DSL ports in a 4-wire or 8-wire bundle is connected through a very noisy cable, it may flap repeatedly. Some data will be dropped every time it flaps.

Suspect mode monitors a link for repeated link drops and will suspend the link for a period of time.

Mode: port dsl <slot> <port>

Step	Command	Purpose
1	device(pf-dsl)[DEFAULT]#suspect-mode [drops <drops> period <period> duration <duration>]	Suspend a port for duration seconds if it drops drops times within period seconds.
2	device(prt-dsl)[0/0]#clear suspect-flag	Re-enable any suspended ports.

Multiport G.SHDSL Devices

The 3310RC and 3310P devices are equipped with multiple G.SHDSL ports that can be configured independently of each other. These devices pass Ethernet traffic directly, and use a sophisticated switching system. The ports on these devices are grouped into slots, with each slot having a set number of ports. (i.e.: The FF3310 has 6 slots with 4 ports.) The **service-mode** command can be used to bond G.S ports within a slot. (The master port in a bond consumes the salve port, i.e.: If dsl 0/0 is configured as **service-mode 4-wire**, dsl 0/1 is no longer configurable independently.)

To make configuring multiple ports easier, the following previously discussed options are available in **profile dsl** instead of **port dsl**, and a **use profile** command is provided in the port: annex-type, mode, payload-rate, snr margin, and tcpam.

Configuring the Profile for the G.SHDSL Connection

In order for the DSL link to connect, Patton devices supporting a 4-wire G.SHDSL card as a CPE client must configure the DSL annex type to match the annex type on the CO side.

Mode: port dsl 0 0

Step	Command	Purpose
1	device(prt-dsl)[0/0]#use profile <name>	Define the dsl profile to use. (Required) Default: DEFAULT

Example: Configuration of the dsl profile

```
profile del DEFAULT
annex-type a-f
mode co
payload-rate 5064
```



```
port dsl 0 0
service-mode 4-wire
use profile DEFAULT
```

VDSL/ADSL Configuration

Introduction

This chapter describes the steps to configure the VDSL/ADSL features of a Trinity device. Once configured, data will be able to be passed through the DSL/Line port to a WAN, usually made available by an Internet Service Provider. However, Internet Service Providers will expect the Trinity device to use certain physical parameters on the line, as well as a specific protocol stack, and we will need to configure the Trinity device to match these requirements. These settings can be obtained from the ISP, and will determine the next steps in configuring the Trinity device.

The physical parameters refer to the specifics of the VDSL and ADSL protocols that we can configure, while the protocol stack refers the type of PTM or ATM WAN connection that is required, where PTM is Packet Transfer Mode and ATM is Asynchronous Transfer Mode. Trinity's support for their respective protocol stacks is listed below:

- PTM
 - IPoE - supported
 - PPPoE - supported
- ATM
 - IPoEoA - supported
 - PPPoEoA - supported
 - IPoA - (available in a future release)
 - PPPoA - (available in a future release)

Configuration Task List

We need to match the ISP's requirements for the physical line properties and protocol stack so the Trinity device can successfully pass data to the ISP's DSLAM (Digital Subscriber Line Access Multiplexer). Once we have that information, the configuration task list is as follows:

- “[Configure Physical Parameters](#)” on page 254—The first step will be configuring the physical parameters of the port, if necessary.
 - **Configure for PTM or ATM:**
 - “[Configure PTM Protocol Stack](#)” on page 254
- or*
- “[Create ATM PVC](#)” on page 256—ATM connections require a bit more information, including the Permanent Virtual Circuit (VCI/VPI) numbers, and any encapsulation and quality-of-service requirements.
 - “[Configure ATM Protocol Stack](#)” on page 256

Configure Physical Parameters

Here we can configure properties that are specific to the VDSL and ADSL protocols. Although the remote DSLAM will require the Trinity device to connect with a particular VDSL/ADSL modulation type, this is determined automatically upon connection, so by default Trinity permits them all (any of G.DMT, ADSL2, ADSL2+, or VDSL2). If necessary, however, they can be individually disabled. Additionally, the specific VDSL2 profiles can be disabled, as well as ADSL's bit-swapping and seamless-rate-adaptation features.

CLI

Mode: configure

Step	Command	Purpose
1	<code>node(cfg)# port dsl slot port</code>	Enters the VDSL/ADSL port configuration mode.
2 (optional)	<code>node(prt-dsl)[slot/port]# [no] dsl-modulation {g.dmt adsl2 adsl2+ vdsl2}</code>	Enables/disables modulation. Default: all enabled.
3 (optional)	<code>node(prt-dsl)[slot/port]# [no] vdsl2-profile {8a 8b 8c 8d 12a 12b 17a 30a}</code>	Enables/disables the VDSL2 profiles. Default: all enabled.
4 (optional)	<code>node(prt-dsl)[slot/port]# [no] adsl-capability {bitswap sra}</code>	Enables/disables bit-swapping and seamless rate adaption (SRA) when in ADSL mode. Default: all enabled.

Configure PTM Protocol Stack

Packet Transfer Mode (PTM) connections include “IPoE” or “PPPoE” protocol stacks.

IPoE

The IPoE type of connection (see [figure 27](#)) bridges Ethernet traffic transparently across a PTM connection.

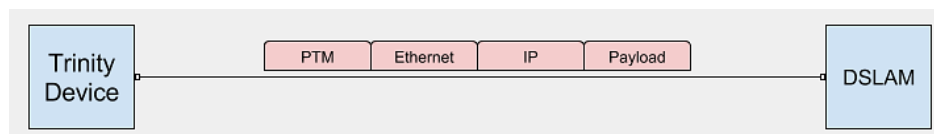


Figure 27. ADSL/VDSL (IPoE)

CLI

Mode: configure

Step	Command	Purpose
1	<code>node(cfg)# port dsl slot port</code>	Enters the VDSL/ADSL port configuration mode.
2	<code>node(prt-dsl)[slot/port]# ptm</code>	Enters the PTM configuration mode.
3	<code>node(ptm)# no shutdown</code>	Enable/disable the PTM interface.
4	<code>node(ptm)# bind interface ctx-name if-name</code>	Bind the PTM interface to an IP address.

Example:

```

context ip ROUTER
interface WAN
  
```

```

        ipaddress WAN dhcp

port dsl 0 0
  ptm
  no shutdown
  bind interface ROUTE

```

PPPoE

The PPPoE type of connection (see [figure 28](#)) encapsulates the IP traffic into a PPP over Ethernet session before it is sent out over PTM.

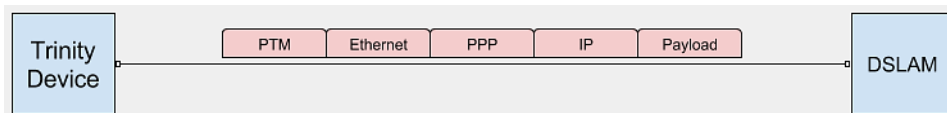


Figure 28. ADSL/VDSL (PPPoE)

CLI

Mode: configure

Step	Command	Purpose
1	<code>node(cfg)# port dsl slot port</code>	Enters the VDSL/ADSL port configuration mode.
2	<code>node(prt-dsl)[slot/port]# ptm</code>	Enters the PTM configuration mode.
3	<code>node(ptm)# no shutdown</code>	Enable/disable the PTM interface.
4	<code>node(ptm)# [no] session pppoe session-name</code>	Creates/deletes a PPPoE session on the PTM interface.
5 (optional)	<code>node(ses-pppoe)[session-name]# access-concentrator ac-name</code>	Active Discovery only accepts PPPoE sessions if the remote peer provides the tag “ac-name” with its Active Discovery Offer. This allows the device to identify the desired remote peer.
6 (optional)	<code>node(ses-pppoe)[session-name]# service service-name</code>	Defines the tag “Service-Name” to be supplied with Active Discovery in order to identify the desired remote peer (check whether the remote peer supports this feature).
7	<code>node(ses-pppoe)[session-name]# bind session ppp session-name</code>	Bind the PPPoE session to a PPP session. The PPP session handles authentication and IP negotiation.

Example:

```

context ip ROUTER
  interface WAN
    ipaddress WAN ipcp

session ppp WAN
  bind interface ROUTER WAN
  no shutdown

```

```

port dsl 0 0
  ptm
  no shutdown
  session pppoe WAN
  bind session ppp WAN

```

Create ATM PVC

An ATM connection is defined by its Permanent Virtual Circuit, which composed of a VPI and a VCI number (Virtual Path Identifier and Virtual Channel Identifier), and an example PVC is 8/35. The PVC can optionally use encapsulation and quality-of-service settings.

CLI

Mode: configure

Step	Command	Purpose
1	<i>node(cfg)# port dsl slot port</i>	Enters the VDSL/ADSL port configuration mode.
2	<i>node(prt-dsl)[slot/port]# atm</i>	Enters the ATM configuration mode.
3	<i>node(prt-dsl)[slot/ port]# [no] pvc vpi vci</i>	Creates the PVC and enters the configuration mode. The “no”-variant deletes the PVC.
4 (optional)	<i>node(pvc)[vpi/ vci]# encapsulation {llc vc}</i>	Optionally sets the PVC’s encapsulation. Select either Logical Link Control (llc) or Virtual Circuit Multiplexing (vc). Default: llc
5 (optional)	<i>node(pvc)[vpi/ vci]# [no] atm-qos {cbr ubr vbr-nrt vbr-rt} pcr pcr cdvt cdvt [scr scr mbs mbs]</i>	Optionally configures the PVC’s Quality of Service. Select one of four bit-rates: Constant (cbr), Unspecified (ubr), Variable Non-Real-Time (vbr-nrt), or Variable Real-Time (vbr-rt). Configure the Peak Cell Rate (pcr), and the Cell Delay Variation Tolerance (cdvt). If using a variable bit rate, the Sustained Cell Rate (scr) and Maximum Burst Size (mbs) must be configured as well. Default: no atm-qos.

Configure ATM Protocol Stack

Asynchronous Transfer Mode (ATM) connections include “IPoEoA” and “PPPoEoA”.

IPoEoA

The IPoEoA type of connection (see [figure 29](#)) bridges Ethernet traffic transparently across an ATM PVC.

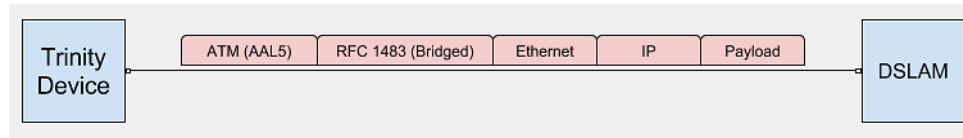


Figure 29. ADSL/VDSL (IPoEoA)

CLI

Mode: configure

Step	Command	Purpose
1	<i>node(cfg)# port dsl slot port</i>	Enters the VDSL/ADSL port configuration mode.
2	<i>node(prt-dsl)[slot/port]# atm</i>	Enters the ATM configuration mode.
3	<i>node(atm)# [no] pvc vpi vci</i>	Creates/deletes the PVC.
4	<i>node(pvc)[vpi/ vci]# [no] ethernet</i>	Enables/disables ATM Ethernet encapsulation.
5	<i>node(pvc-eth)[vpi/ vci]# [no] shutdown</i>	Enables/disables the PVC.
6	<i>node(pvc-eth)[vpi/ vci]# bind interface ctx-name if-name</i>	Bind the PVC to an IP interface.

Example:

```
context ip ROUTER
interface WAN
  ipaddress WAN dhcp

port dsl 0 0
  atm
    pvc 8 35
      ethernet
        no shutdown
        bind interface ROUTER WAN
```

PPPoEoA

The PPPoEoA type of connection (see [figure 30](#)) creates a PPP session over the ATM PVC.

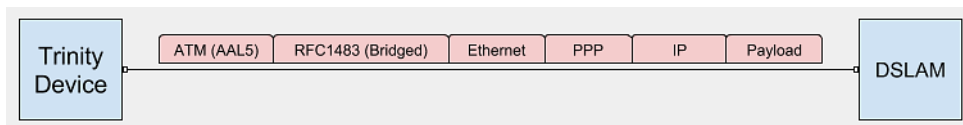


Figure 30. ADSL/VDSL (PPPoEoA)

CLI

Mode: configure

Step	Command	Purpose
1	<code>node(cfg)# port dsl slot port</code>	Enters the VDSL/ADSL port configuration mode.
2	<code>node(prt-dsl)[slot/ port]# atm</code>	Enters the ATM configuration mode.
3	<code>node(atm)# [no] pvc vpi vci</code>	Creates/deletes the PVC.
4	<code>node(pvc)[vpi/ vci]# [no] ethernet</code>	Creates/deletes a PPPoE session on the PTM interface.
5 (optional)	<code>node(ses-pppoe)[session-name]# access-concentrator ac-name</code>	Active Discovery only accepts PPPoE sessions if the remote peer provides the tag “ac-name” with its Active Discovery Offer. This allows the device to identify the desired remote peer.
6 (optional)	<code>node(ses-pppoe)[session-name]# service service-name</code>	Defines the tag “Service-Name” to be supplied with Active Discovery in order to identify the desired remote peer (check whether the remote peer supports this feature).
7	<code>node(ses-pppoe)[session-name]# bind session ppp session-name</code>	Bind the PPPoE session to a PPP session. The PPP session handles authentication and IP negotiation.

Example:

```

context ip ROUTER
  interface WAN
    ipaddress WAN ipcp

  session ppp WAN
    bind interface ROUTER WAN
    no shutdown

  port dsl 0 0
    atm
      pvc 8 35
        ethernet
          no shutdown
          session pppoe WAN
          bind session ppp WAN

```

Troubleshooting DSL Connections

Checking DSL connection status with show command

Details about the current state of the DSL connection can be inspected with the *show* command. For example, to see the operational state of the port itself, or the status of the ATM or PTM interface.

Step	Command	Purpose
1	<code>node(cfg)# show port dsl slot port</code>	Show status and configuration details of the dsl port.

Link State

Verify that the DSL link is established (status LED is continuously on)

Debugging the DSL connection with the DSL debug command.

Do the following to enable the DSL debugs:

Step	Command	Purpose
1	<code>debug dsl [{detail <level> full-detail}]</code>	Enables the DSL debugs.

Chapter 18 **Context Bridge**

Chapter contents

Introduction	261
Bridge group Configuration Task List	261

Introduction

This chapter outlines the Context Bridge. You will obtain a fundamental understanding of how to setup your Patton Trinity Embedded device for bridge configurations. The Context Bridge in Trinity is a high level conceptual entity that is responsible for the management of relaying and filtering of frames from at least two physical Ethernet ports found at the MAC layer of the OSI model.

Bridge group Configuration Task List

The following sections describe how to configure/use the Bridging component:

- Creating a bridge group
- Setting various bridge options
- Bind resources to the bridge-group
- Enable filters on the bridge-group
- Set STP options
- Configure VLANs

Mode: context bridge

Table 19. Creating Bridge Group

Step	Command	Purpose
1	node# context bridge	Enters the Bridge configuration context.
2	node~(ctx-br)# bridge-group <name>	Creates a new bridge group with the name <name>.

Table 20. Setting Various Bridge Options

Step	Command	Purpose
1	node~(ctx-br)[TEST]# ageing <seconds>	Sets the desired ageing options. Default is 299.95secs. The ageing is how long a host information will be kept in the bridge database before updating it.
2	node~(ctx-br)[TEST]# arp	Enables ARP protocol on the bridge port
3	node~(ctx-br)[TEST]# multicast	Enables IP Multicast on the bridge port
4	node~(ctx-br)[TEST]# bind	Binds resources to the bridge group
5	node~(ctx-br)[TEST]# filter	Enable various filters on the bridge group
6	node~(ctx-br)[TEST]# stp	Set STP Options
7	node~(ctx-br)[TEST]# vlan <id>	Enters VLAN Configuration mode
8	node~(ctx-br)[TEST]# settap	Taps the interface to mirror communications useful in debugging in situations like using Wireshark.

Table 21. Bind Resources to the Bridge-Group

Step	Command	Purpose
1	node~(ctx-br)[TEST]# bind interface <IP_Interface>	Binds this bridge-group to an IP Interface. In other words, it allows the bridge-group to have the IP settings. Remember a bridge-group is just like an Ethernet port for the most part.

Binding an IP Interface to a bridge-group works identical to binding an interface to an Ethernet port.

Table 22. Enable Filters on the Bridge-Group

Step	Command	Purpose
1	node~(ctx-br)[TEST]# filter mac	Configures IP Table rules based on mac address.
2	node~(ctx-br)[TEST]# filter stp	Configures STP Filters based on interfaces.

Table 23. Set STP Options

Step	Command	Purpose
1	node~(ctx-br)[TEST]# stp bridgeprio <prio>	Set bridge priority [0-65535] (Default: 32768).
2	node~(ctx-br)[TEST]# stp fwdelay <seconds>	Set bridge forwarding delay in seconds (Default:15s).
3	node~(ctx-br)[TEST]# stp hello <seconds>	Set hello time interval in seconds (Default: 20s).
4	node~(ctx-br)[TEST]# stp maxage <seconds>	Set maximum hello message age in seconds (Default: 20s).

All VLAN options are identical through VLAN configuration options for normal interfaces. See “[VLAN \(802.1p/Q\)](#)” on page 218 for a detailed guide.

Table 24. Configure VLANs

Step	Command	Purpose
1	node~(ctx-br)[TEST]# vlan <id>	Enter VLAN Confirmation mode <id> = [1..4094]
2	node~(vlan)[br-TEST.1]# arp	Enables ARP.
3	node~(vlan)[br-TEST.1]# mtu	Sets MTU.
4	node~(vlan)[br-TEST.1]# multicast	Enables multicast.
5	node~(vlan)[br-TEST.1]# map	Maps VLAN class of service (CoS) value to internal traffic class and vice-versa
6	node~(vlan)[br-TEST.1]# bind	Binds a resources to a VLAN. You can bind this VLAN to an IP Interface or you can bind it to a bridge-group.

VLANs take one port and split it into several logical Ethernet port, therefore binding resources to a VLAN is the same as binding a resource to an Ethernet Port. See “[IP Context Overview](#)” on page 280 for details.

Mode: administrator access

Table 25. Show Bridge-Group Configuration

Step	Command	Purpose
1	node# show bridge <name>	Shows bridge-group configuration for <name>.

Chapter 19 **Spanning Tree Configuration**

Chapter contents

Introduction.....	265
Spanning Tree Configuration Task List.....	265
Configuring Global Spanning Tree Parameters	266
Configuring Per-tree Spanning Tree Parameters	266
Configuring Per-port/Per-tree Spanning Tree Parameters	267
Enabling Spanning Tree on a Port	267
Debugging Spanning Tree	267
Spanning Tree Configuration Example	268

Introduction

Devices that have Ethernet switch ports support the following spanning tree protocols:

- Classic spanning tree (STP)
- Rapid spanning tree (RSTP) as defined in IEEE 802.1w
- Multiple spanning tree (MSTP) as defined in IEEE 802.1s

Note This chapter is intended to provide general, non-product specific information on classic STP, RSTP and MSTP, which is widely available online and in print.

Each of the spanning tree protocols is used to prevent loops in a layer 2 network. Loops must not be present in a layer 2 network because broadcast packets will be continuously rebroadcast and use up all of the network's bandwidth.

However, having multiple connections between switches is desirable because it allows for redundancy. If one of its connections goes down, it may use one of its other connections to reach the other switch.

All of the spanning tree protocols allow for redundancy by allowing multiple connections between switches. One of the connections is put in the *forwarding* state, while the others are put in the *blocking* state. If the connection in the *forwarding* state goes down, one of the connections in the *blocking* state transitions to the *forwarding* state. Rapid spanning tree protocol (RSTP) improves on classic spanning tree by transitioning faster to the *forwarding* state, resulting in less downtime.

In addition, multiple spanning tree protocol (MSTP) allows for load balancing by mapping one set of VLANs to one spanning tree instance, another set of VLANs to another spanning tree instance, and so on. Each of the spanning tree instances may have different ports that are *forwarding*. So VLANs 100 and 101 may be forwarded out of port 1, while VLANs 200 and 201 are forwarded out of port 2. This makes better use of the available bandwidth. Then, if port 2 goes down, VLANs 200 and 201 will be forwarded out of port 1, in addition to VLANs 100 and 101.

Spanning Tree Configuration Task List

By default, the device is configured to run the multiple spanning tree protocol version (MSTP). This is the recommended configuration because MSTP is backwards compatible with the other spanning tree protocol versions. That is, if another device that only supports STP or RSTP is connected to one of this device's ports, this device will automatically fall back to STP or RSTP.

By default, the MST configuration name is empty and the revision is zero. These must be configured or else the device will run the rapid spanning tree protocol version (RSTP). All devices in a MST region must have the same MST configuration name and revision.

By default, there is only one configured spanning tree instance, the common and internal spanning tree (CIST), with all VLANs mapped to it. All devices in a MST region must have the same VLAN to spanning tree instance mapping.

By default, the spanning tree protocol is disabled on all ports. It must be enabled on a per-port basis.

Configuring Global Spanning Tree Parameters

Mode: Configure

Step	Command	Purpose
1	device(cfg)#switch spanning tree	Enters global spanning tree configuration mode.
2	device(stp)#protocol-version {stp rstp mstp}	Configures the spanning tree protocol version.
3	device(stp)#tx-hold-count <i>count</i>	Optional: Configures the transmit hold count. This should normally be left at the default value.
4	device(stp)#mst-config name <i>name</i>	N/A for STP and RSTP: Configures the MST configuration name, a string up to 32 characters long.
5	device(stp)#mst-config revision <i>revision</i>	N/A for STP and RSTP: Configures the MST configuration revision, a number 0 to 65535, inclusive.

Configuring Per-tree Spanning Tree Parameters

Mode: Global spanning tree configuration

Step	Command	Purpose
1	device(stp)#tree {cist msti <i>instance-id</i> }	Creates a new spanning tree instance if it doesn't exist, and enters its configuration mode. For STP and RSTP, only the CIST is applicable. For MSTP, <i>instance-id</i> may be 1 to 64, inclusive. The no form of this command may be used to destroy any instance other than the CIST.
2	device(stp) [instance-id]#vlan <i>vlan-id</i>	Maps a VLAN to this spanning tree instance. For STP and RSTP, this does not apply.
3	device(stp) [instance-id]#priority <i>priority</i>	Configures this device's priority in this spanning tree instance's root bridge. The device that you want to be the root must be given the lowest priority.
4	device(stp) [instance-id]#max-age <i>seconds</i>	This only applies to the CIST; and configures the max age in seconds. A Bridge Protocol Data Unit (BPDU) will no longer be forwarded after its max age expires. It must be between 6 and 40, inclusive, and also be less than or equal to $2 * (\text{forward-delay} - 1)$. This should normally be left at the default value.
5	device(stp) [instance-id]#forward-delay <i>seconds</i>	This only applies to the CIST; and configures the forward delay in seconds. A port will wait this long before entering the forwarding state. It must be between 4 and 30, inclusive, and also be greater than to $(\text{max-age} / 2) + 1$. This should normally be left at the default value.
6	device(stp) [instance-id]#max-hops <i>count</i>	Configures the max hops. This is the maximum numbers of bridges a BPDU may be forwarded to before it expires.

Configuring Per-port/Per-tree Spanning Tree Parameters

Mode: Switch port configuration

Step	Command	Purpose
1	device(switch-port-type) [<i>slot/port</i>] #spanning-tree { <i>cist</i> <i>msti instance-id</i> }	Enters the per-port/per-tree spanning tree configuration mode.
2	device(stp-type) [<i>slot/port</i>][<i>instance-id</i>] #priority <i>priority</i>	Configures this port's priority in this spanning tree instance, a number 0 to 240, inclusive, in steps of 16. This is used as a tie-breaker if two ports can both reach the root bridge with the same path cost. The port with the lower priority will be the root port, and the other port will be the alternate port.

Enabling Spanning Tree on a Port

Mode: Switch port configuration

Step	Command	Purpose
1	device(switch-port-type) [<i>slot/port</i>] #spanning-tree	Enables the spanning tree protocol on the given port. The no form of the command disables the spanning tree protocol on this port.

Debugging Spanning Tree

The **show switch spanning-tree** command may be used to show the spanning tree configuration as well as the current topology.

Mode: Operator execution

Step	Command	Purpose
1	device#show switch spanning-tree	Show the spanning tree configuration and the current topology.

Additionally, the **trace ethsw-stp** command may be used to log information about the spanning tree protocol operation.

Mode: Operator execution

Step	Command	Purpose
1	device#trace ethsw-stp debug detail <i>level</i>	Enables logging of the spanning tree protocol operation; <i>level</i> may be 1 (topology changes), 2 (state machine transitions), or 3 (BPDU send and received).

Spanning Tree Configuration Example

Example: Configure the FF3310RC for the following network:

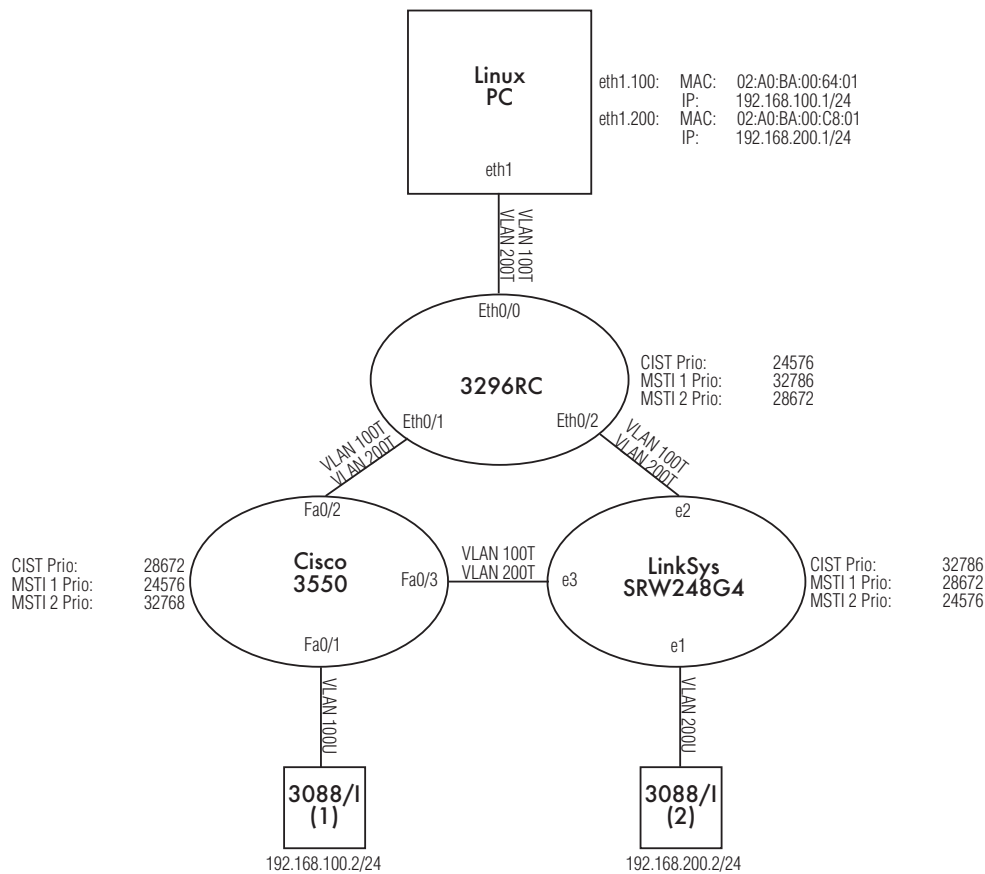


Figure 31. Spanning tree configuration

In this network, there are three switches, the FF3310RC, the Cisco 3550, and the LinkSys SRW248G4. The FF3310RC is the CIST root, the Cisco 3550 is the MSTI 1 root, and the LinkSys SRW248G4 is the MSTI 2 root. Any one of the three links may go down and the Linux PC will still be able to access both 3088/Is.

```
device(cfg)#switch spanning-tree
device(stp)#mst-config name REGION_A
device(stp)#tree cist
device(stp)[0]#priority 24576
device(stp)[0]#exit
device(stp)#tree msti 1
device(stp)[1]#vlan 100
device(stp)[1]#exit
device(stp)#tree msti 2
device(stp)[2]#priority 28672
device(stp)[2]#vlan 200
device(stp)[2]#exit
device(stp)#exit
device(cfg)#port ethernet 0 0
```



```
device(prt-eth)[0/0]#no shutdown
device(prt-eth)[0/0]#switch
device(switch-port-ethernet)[0/0]#vlan 100 tagged
device(switch-port-ethernet)[0/0]#vlan 200 tagged
device(switch-port-ethernet)[0/0]#exit
device(prt-eth)[0/0]#exit
device(cfg)#port ethernet 0 1
device(prt-eth)[0/1]#no shutdown
device(prt-eth)[0/1]#switch
device(switch-port-ethernet)[0/1]#vlan 100 tagged
device(switch-port-ethernet)[0/1]#vlan 200 tagged
device(switch-port-ethernet)[0/1]#exit
device(prt-eth)[0/1]#spanning-tree
device(prt-eth)[0/1]#exit
device(cfg)#port ethernet 0 2
device(prt-eth)[0/2]#no shutdown
device(prt-eth)[0/2]#switch
device(switch-port-ethernet)[0/1]#vlan 100 tagged
device(switch-port-ethernet)[0/1]#vlan 200 tagged
device(switch-port-ethernet)[0/1]#exit
device(prt-eth)[0/2]#spanning-tree
device(prt-eth)[0/2]#exit
device(cfg)#
```

Chapter 20 **PPP Configuration**

Chapter contents

Introduction	271
PPP Configuration Task List	272
Creating an IP Interface for PPP	272
Creating a PPP Session	273
Configuring a PPPoE Session	275
Creating a PPP Profile	275
Displaying PPP Configuration Information	277
Debugging PPP	277
Sample Configurations	278
PPP Over Ethernet (PPPoE)	278
Without authentication, encapsulation multi, with NAPT	278
With authentication, encapsulation PPPoE	279

Introduction

This chapter describes how to configure the point-to-point protocol over different link layers.

The Point-to-Point Protocol (PPP) provides a standard method for transporting multi-protocol datagrams over point-to-point links as defined by the RFC1661 etc. Trinity offers PPP over the following link layers:

- PPP over Ethernet (PPPoE)

Figure 32 shows the elements involved in the configuration of PPP. The elements required to configure PPP over the Ethernet are located in the upper left corner of the figure.

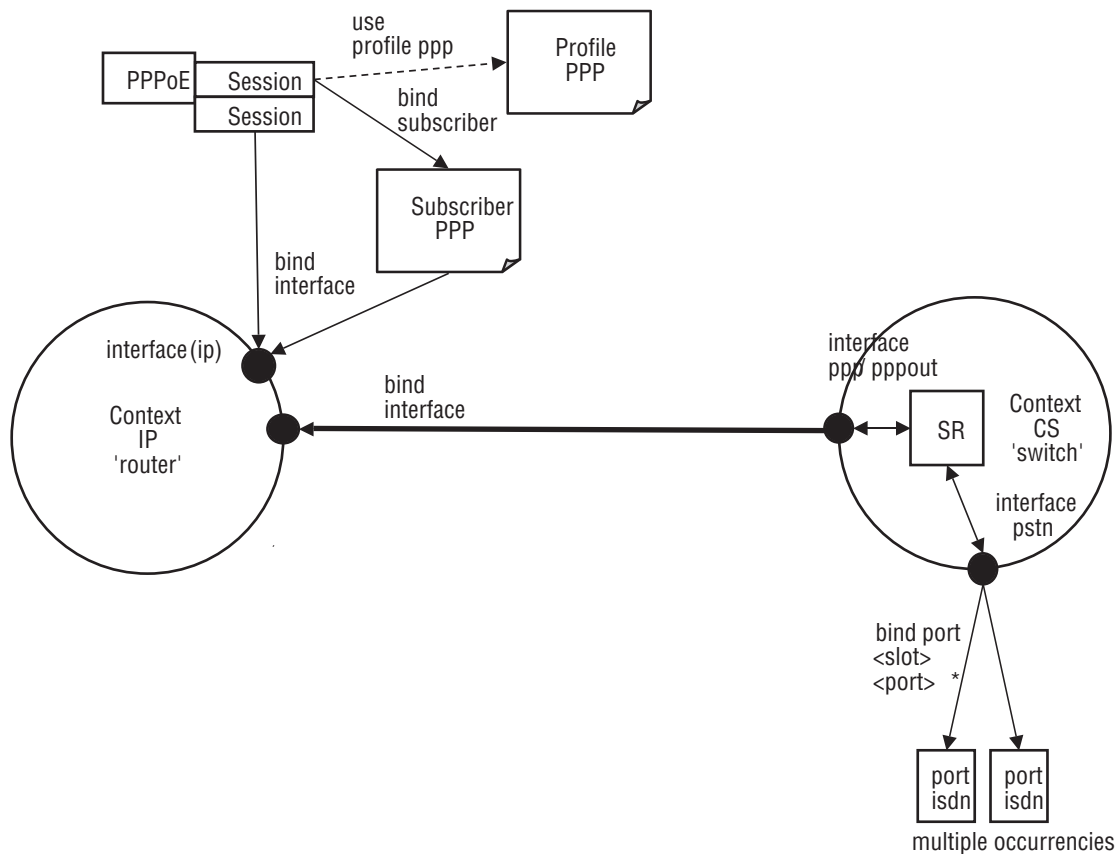


Figure 32. PPP configuration overview

Since the purpose of PPP is providing IP connectivity over different types of link layers, all PPP configuration elements connect to the IP context through an IP interface. This connection is relayed via PPP session.

For PPP over Ethernet, a PPPoE session must be configured on the respective Ethernet port. It is possible to set-up several (limited by the available memory) PPPoE sessions on the same Ethernet port, each session with its own IP interface. In addition to these PPPoE sessions, pure IP traffic can run concurrently over the same Ethernet port. This is achieved by binding the Ethernet port directly to an IP interface.

PPP Configuration Task List

To configure PPP, perform the following tasks:

- Creating an IP interface for PPP
- Configuring for IP address auto-configuration from PPP (see page 275)
- Creating a PPP session (for authentication) (see page 273)
- Configuring a PPPoE session (see page 275)
- Creating a PPP profile (see page 275)
- Displaying PPP configuration information (see page 277)
- Debugging PPP (see page 277)

Creating an IP Interface for PPP

An IP interface is required to link a PPP connection to the IP context. The IP interface must apply a network address port translation (NAPT) if the IP addresses on the LAN shall be private and hidden behind a public IP address (see [Chapter 30, “NAT/NAPT Configuration”](#) for more information about NAPT).

This procedure describes how to create an IP interface for PPP.

Mode: Context IP

Step	Command	Purpose
1	<code>node(ctx-ip)[ROUTER]#interface <i>name</i></code>	Creates the new interface <i>name</i> , which represents an IP interface.

Step	Command	Purpose
2	device(if-ip)[<name>]#ipaddress [<label>] ipcp [request] [<i>ip-address</i>] [peer [request] [<i>peer-ip-address</i>]] [ignore dns]	<ul style="list-style-type: none"> • If no address label is specified, the name of the interface will be taken. Negotiate the IP address with the PPP remote peer using PPP's IP control protocol (IPCP). • If <i>ip-address</i> is not specified, we will require the PPP remote peer to offer us our local IP address. • If request <i>ip-address</i> is specified, we will request the PPP remote peer to offer us <i>ip-address</i> as our local IP address, but we will accept another local IP address, if it rejects our request. • If <i>ip-address</i> is specified, then we will request the PPP remote peer to offer us <i>ip-address</i> as our local IP address, and we will fail to connect unless it accepts our request. • If <i>peer-ip-address</i> is not specified, we will require the PPP remote peer to assign its own local IP address and tell us what it is. • If request <i>peer-ip-address</i> is specified, we will request the PPP remote peer to assign itself <i>peer-ip-address</i> as its own local IP address, but we will accept another peer IP address if it rejects our request. • If <i>peer-ip-address</i> is specified, then we will require the PPP remote peer to assign itself <i>peer-ip-address</i> as its own local IP address, and we will fail to connect unless it accepts our request. • By default, we will request up to two DNS servers from the PPP remote peer, but we will not if ignore dns is specified.
3 (optional)	node(if-ip)[name]#use profile napt <i>name</i>	Assigns the NAPT profile <i>name</i> to applied to this IP interface.

Example: Create an IP interface for PPP

The following procedure creates an IP interface that can be used for all three types of link layers. The command lines **tcp adjust-mss** only apply to Ethernet link layers.

```
node(cfg)#context ip ROUTER
node(ctx-ip)[ROUTER]#interface PPP_INTERFACE
node(if-ip)[ROUTER.PPP_INT~]#ipaddress IPCP
```

Creating a PPP Session

One or more PPP session will be configured if either PPP peer requires authentication. This procedure describes how to create a PPP session:

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg) # session ppp<session-name></code>	Creates the new session <i>name</i> .
2	<code>node(session-ppp)[name]# [no] authentication { (chap pap) {chap pap} }</code>	Defines the authentication protocol to be used, PAP and/or CHAP.
3 (optional)	<code>node(session-ppp)[name]# [no] identification {outbound inbound} user [password password]</code>	<p>Sets the credentials to be provided during the authentication procedure:</p> <ul style="list-style-type: none"> the user name: <i>user</i> the password: <i>password</i> <p>The keywords 'inbound' and 'outbound' define the direction of authentication:</p> <ul style="list-style-type: none"> 'inbound': The local peer checks the credentials that the remote peer sends. 'outbound': The local peer sends its credentials if the remote peer requests them. <p>The following restrictions apply to the direction of authentication:</p> <ul style="list-style-type: none"> PPP over Ethernet: 'outbound' only
4	<code>node(session-ppp)[name]# [no] bind interface [ROUTER] interface</code>	Binds the session to the IP interface to be used for this PPP connection. The IP interface must already exist and shall have the configuration as outlined in section “Creating an IP Interface for PPP” on page 272.
5 (optional)	<code>node (session-ppp)[name]# [no] use profile ppp <name></code>	Assigns a PPP profile other than the default profile to this PPP session.
6	<code>node(session-ppp)[name]#[no] shutdown</code>	<p>Initiates the establishment of the PPP session and the PPP connection.</p> <p>Note A PPP link, such as a PPPoE session must be bind to this PPP session before it can initiate.</p>

Example: Create a PPP session

The procedure below creates a PPP session for a PAP authentication with some Internet Service Provider.

```
node(cfg)#session ppp JOE_EXAMPLE
node(session-ppp)[JOE_EXA~]#authentication pap
node(session-ppp)[JOE_EXA~]#identification outbound joeexample@isp.com password
blue4you
node(session-ppp)[JOE_EXA~]#bind interface ROUTER PPP_INTERFACE
node(session-ppp)[JOE_EXA~]# no shutdown
```

Configuring a PPPoE Session

PPP can run over Ethernet (PPPoE). The *active discovery* protocol identifies the PPP remote peer on the Ethernet and establishes a PPPoE session with it. The PPPoE session provides a logical point-to-point link that runs PPP as if it was a physical point-to-point link (e.g. a serial link).

This procedure describes how to configure an Ethernet port and a session for PPPoE:

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg) #port ethernet slot port</code>	Enters Ethernet port configuration mode for the interface on <i>slot</i> and <i>port</i>
2 (optional)	<code>node (prt-eth)[slot/port]# [no] bind interface name [ROUTER]</code>	Binds the Ethernet port to the IP interface to be used for the direct IP traffic.
3	<code>node(prt-eth)[slot/port]#[no] shutdown</code>	Enables the ethernet port
4	<code>node(prt-eth)[slot/port]#session pppoe name</code>	Creates PPPoE session with the name: <i>name</i>
5	<code>node(session)[name]# [no] bind session ppp name</code>	Binds the PPPoE session to the PPP session <i>name</i> .
6 (optional)	<code>node(session)[name]#service Service-Name</code>	Defines the tag 'Service-Name' to be supplied with Active Discovery in order to identify the desired remote peer (check whether the remote peer supports this feature)
7 (optional)	<code>node(session)[name]#access-concentrator AC-Name</code>	The Active Discovery only accepts the PPPoE session if the remote peer provides tag 'AC-Name' with its Active Discovery Offer as specified. This allows to identify the desired remote peer

Example: Configure a PPPoE session

The procedure below configures a PPPoE session for the connection to a DSL provider using the credentials specified in the PPP session profile above.

```
node(cfg)#port ethernet 0 0
node(prt-eth)[0/0]#no shutdown
node(prt-eth)[0/0]#session pppoe green
node(session)[GREEN]#bind session ppp JOE_EXAMPLE
```

Creating a PPP Profile

A PPP profile allows to adjust additional PPP parameters like the maximum transmit unit (MTU) and maximum receive unit (MRU). Only the most important parameters are listed here.

The profile *DEFAULT* is always present and supplies the parameters if no other profile has been created or a profile cannot be used with a certain type of PPP connection. Profiles created by the user can only be used with PPP over Ethernet connections. For all other types of PPP connections the default profile applies.

The procedure below describes how to create a PPP profile or to modify the default PPP profile.

Mode: Configure

Step	Command	Purpose
1	<i>node(cfg) #[no] profile ppp { name DEFAULT }</i>	Creates the new PPP profile <i>name</i> and enters the PPP profile configuration. The profile 'DEFAULT' already exists.
2 (optional)	<i>node(pf-ppp)[name]#mtu min max</i>	Defines the minimum and maximum size of IP packets (in Bytes) allowed on the outbound PPP connection. Outbound packets larger than the maximum size are fragmented into smaller ones if allowed. The default value is 1492 Bytes. On the IP interface over which the PPP connection runs, the minimum of the IP interface MTU and PPP MTU applies.
3 (optional)	<i>node(pf-ppp)[name]#mru max</i>	Defines the minimum and maximum size of IP packets (in Bytes) allowed on the inbound PPP connection. The default value is 1492 Bytes. Inbound packets larger than the maximum size are fragmented into smaller ones if allowed. The default value is 1492 Bytes.
4 (optional)	<i>node(pf-ppp)[name]#[no] van-jacobson {compression decompression} max-slots max-slots</i>	Allows PPP to use Van Jacobson header compression for TCP packets. Only the negotiation between the PPP peers determines whether this header compression is really used. <i>max-slots</i> determines the maximum number of concurrent TCP sessions for which header compression shall be done. The default is 31.

Example: Create a PPP profile

The procedure below creates a PPP profile, sets some of its parameters, and assigns it to a PPPoE session.

```
node(cfg)#profile ppp PPPoE
node(pf-ppp)[PPPoE]#mtu min 68 max 1492
node(pf-ppp)[PPPoE]#mru min 68 max 1492
node(pf-ppp)[PPPoE]#van-jacobson
node(pppoe)[0/0]#session ppp JOE_EXAMPLE
node(session-ppp)[JOE_EXA~]#use profile ppp PPPoE
```


Displaying PPP Configuration Information

This section shows how to display and verify the PPP configuration information.

Mode: Configure

Step	Command	Purpose
1	node(cfg) #show running-config	Gives the best overview of all PPP related configuration information. The following parts of interest are: <ul style="list-style-type: none"> • profile ppp DEFAULT • profile ppp <i>name</i> • interface <i>name</i> • session ppp <i>name</i> • port Ethernet <i>slot port</i> • session pppoe <i>name</i>
2	node(cfg) #show session ppp [<i>name</i>]	Displays configuration information of the PPP session <i>name</i> or of all PPP sessions.

Example: Display PPP session configuration information

```
node#show session ppp JOE_EXAMPLE

ppp JOE_EXAMPLE
=====
Administrative State : Up
Bound IP Interface   : ROUTER.PPP_INTERFACE
IP Addresses         : IPCP (IPCP): up
                     : Requested: (none)
                     : Operational: 10.67.15.1/32 peer 10.0.0.1
Bound Bridge         : (none)
Operational State    : Up
Hardware Address     : 00:00:0c:16:3b:43
MTU                  :
ARP                  : enabled
Multicast             : enabled
Rx Statistics        : 1058 bytes in 16 packets
                     : 0 errors 0 drops, 0 overruns
                     : 0 multicast packets
Tx Statistics        : 1058 bytes in 16 packets
                     : 0 errors 0 drops, 0 collisions 0 carrier errors
```

Debugging PPP

A set of commands is available to check the status of the PPP connection and the PPPoE session. Furthermore, two debug monitors help to analyze the dynamic behavior. The commands are listed in the order which you should follow in case you encounter problems with PPP. This procedure describes how to display PPP configuration information:

Mode: Configure

Step	Command	Purpose
1	<i>node(cfg) #show pppoe [name]</i>	Displays configuration information of the PPPoE session(s). <i>level</i> specifies to level of details displayed (1..4, default is 1).
2	<i>node(cfg) #show port ethernet slot port</i>	Displays status and configuration information of the Ethernet/serial port over which a PPP connection/PPPoE sessions runs. Check whether operation state of the port is 'Up'.
3	<i>node(cfg) # [no] trace ppp debug detail level</i>	Enables all or a particular PPP debug monitor.
4	<i>node(cfg) # [no] trace pppoe debug detail level</i>	Enables all or a particular PPPoE debug monitor.

Example: Display PPPoE information

```
GREEN
=====
Interface           : ethernet 0 0
Service             :
Access Concentrator :
PPP Session         : JOE_EXAMPLE
```

Sample Configurations

PPP Over Ethernet (PPPoE)

Without authentication, encapsulation multi, with NAPT

```
profile napt WAN

context ip ROUTER

    interface NORMAL_IP_INTERFACE
        ipaddress 172.16.1.1 255.255.0.0

    interface PPP_INTERFACE
        ipaddress IPCP
    use profile napt WAN

context ip ROUTER
routing-table DEFAULT
route 0.0.0.0 0.0.0.0 interface PPP INTERFACE metric 0

session ppp NO AUTHENTICATION
bind interface ROUTER PPP INTERFACE
no shutdown

port ethernet 0 0
bind interface normal_ip_interface
```

```
no shutdown

session pppoe GEEEN
bind session ppp NO AUTHENTICATION
```

With authentication, encapsulation PPPoE

```
context ip ROUTER

interface PPP_INTERFACE
  ipaddress IPCP

session ppp JOE_EXAMPLE
authentication pap
  identification outbound <user> password <password>
bind interface ROUTER PPP_INTERFACE

port ethernet 0 0
no shutdown

session pppoe GREEN
  bind session ppp JOE_EXAMPLE
```

Chapter 21 IP Context Overview

Chapter contents

Introduction	281
Packet Processing in the IP Context	282
Classifier	284
Network Address Port Translation (NAPT)	284
Routing-table Selection	284
Access Control Lists (ACL)	284
Routing	284
Packet Processing To/From Local Applications	285
IP Context Overview Configuration Task List.....	285
Planning Your IP Configuration.....	286
IP Interface Related Information	286
QoS Related Information	286
Configuring Physical Ports	286
Creating and Configuring IP Interfaces	287
Configuring Packet Classification	287
Configuring Network Address Port Translation (NAPT)	287
Configuring Static IP Routing	287
Configuring Access Control Lists (ACL)	288
Configuring Quality of Service (QoS)	288

Introduction

This chapter outlines the Trinity *Internet protocol (IP) context* and its related components. You will get the fundamental understanding on how to set up your Patton device to make use of IP related services.

The following sections describe the configuration steps necessary to put together certain IP services and the references to the related chapters that explain the issue in more details.

Trinity also supports Fast-Path routing, which can be found in [Chapter 29](#) on page 369.

To understand the information given in the following chapters, carefully read to the end of the current chapter. Before proceeding, make sure that you feel comfortable with the underlying Trinity configuration concept by reading Chapter 2, “[Configuration Concepts](#)” on page 57.

The IP context in Trinity is a high level, conceptual entity that is responsible for all IP-related protocols and services for data and voice. The IP context performs much of the same functions as a standalone IP router, and since every context is defined by a name, the IP context is named *ROUTER* by default.

In [figure 33](#) below, the IP context with all its related elements is contained within the area on the left, which has a gray fill (find a short description of those elements below). The right side displays the related CS context, which communicates with the IP context via gateways. Since the CS context and its related components are not the subject of this chapter, they are illustrated in [figure 33](#) with gray lines instead of black ones.

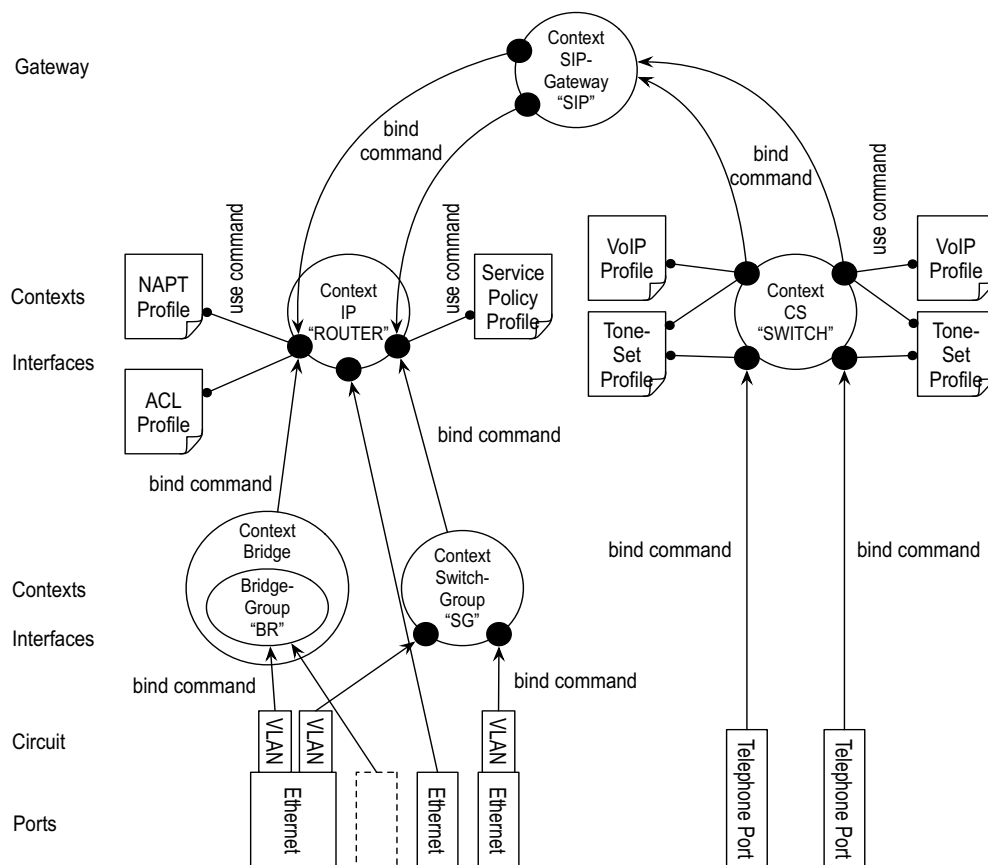


Figure 33. IP context and related elements

The IP context contains the following entities:

- Routing tables
- Logical IP interface
- Links to service profiles

Since the IP context represents a virtual IP4 and IPv6 dual-stack router, it contains up to 251 routing tables for static routes (not depicted in [figure 33](#) on page 281). The routing tables decide whether received packets are delivered to a local application (example, CLI, web server, SIP gateway) or routed via another IP interface to a remote network host.

The IP context may contain an arbitrary number of logical interfaces. Unlike other operating systems where a network interface is identical to a physical port, we distinguish physical ports from logical interfaces. A logical interface contains all IP-related configuration parameters that are common to all ports, such as the IP address, for example. As depicted in [figure 33](#) on page 281, a physical port or circuit is bound bottom-up to one logical IP interface. Hence, each IP interface reflects the IP-protocol of a physical port or circuit.

Applications such as SIP gateways may also be bound to an IP interface. A top-down binding defines over which IP interface (and hence over which physical port or circuit) an application communicates.

Packet Processing in the IP Context

Several IP service profiles can be assigned to the individual logical interfaces in the context (see [figure 33](#)). These profiles control the flow of packets through the router. They classify packet streams, control which packets may enter/leave the device via Access Control Lists (ACL), perform Network and Port Address Translation (NAPT) and deal with Quality-of-Service (QoS) information in packet headers.

Note that there is a different packet-processing chain for each interface depending on its configuration, i.e., each interface maintains its own configuration of how the packets are classified, a different ACL, etc. However, to make having the same configuration on multiple interfaces easier, we moved the configuration parameters to profiles. The **use** command attaches a profile to an interface, such that the same profile can be used by different interfaces.

Figure 34 shows the journey of a packet through the IP context and the order in which the attached profiles process the packet.

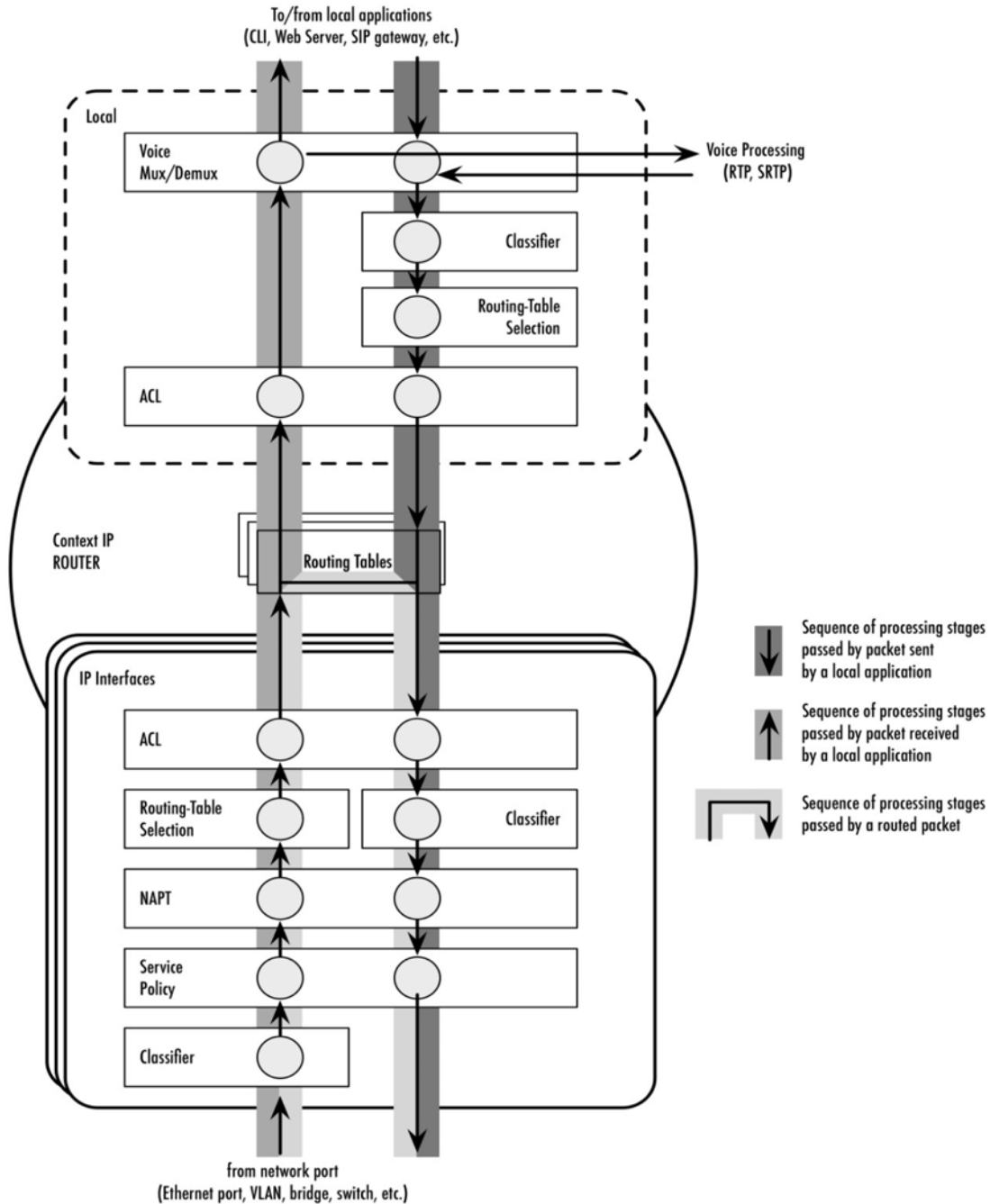


Figure 34. Processing order of IP services attached to an IP interface

Classifier

The classifier is the first profile that inspects an incoming packet. The classifier assigns a traffic class to each packet. You can think of the traffic-class as if every packet in the router has a tag attached to it, on which the classification can be noted. The traffic-class tags exist only inside the router, but layer 2 priority bits (802.1pq class-of-service) and IP header type-of-service bits (TOS field) can be used to mark a specific packet type for the other network devices. By default the traffic-class tag is *DEFAULT*.

A powerful packet-matching filter in the classifier profile lets you inspect any combination of IP, UDP, TCP or ICMP header fields and assign a traffic-class to the matching packet flow. For example, you may configure to tag all UDP packets to a destination port between 5000 and 8000, and shorter than 500 bytes with the traffic-class *VOICE*. The traffic-class tag can later be used in other IP service profiles, e.g., to filter packets in the ACL or to do policy routing by selecting a routing-table based on the traffic-class.

Network Address Port Translation (NAPT)

After classification is done, the packet is handed over to the NAPT profile-if one is used on the current interface. Network Address Port Translation (NAPT), which is an extension to NAT, uses TCP/UDP ports in addition to network addresses (IP addresses) to map multiple private network addresses to a single outside address. Thus the NAPT profile may change the destination address and port of an incoming packet.

Routing-table Selection

You may configure policy routing by selecting a different routing table based on some header fields of the incoming packet. You may also use the traffic-class (tagged before in the Classifier) to make a routing-table decision. For example, you may direct all packets tagged with the *VOICE* traffic-class to a separate routing table while processing the other traffic with the default routing table.

Note The routing-table selection for an incoming packet is performed after NAPT, i.e., you will see the translated (private) addresses and ports

Access Control Lists (ACL)

An access control list is a sequential collection of permit and deny conditions that apply to packets on a certain interface. You can use the same packet-matching mechanism as in the classifier and the routing-table selection to decide whether the specified packet flow is permitted to enter the router or is rejected.

The ACL filter is passed after the routing decision has been made. This allows you to apply an ACL to an input-output interface pair. For example, you may use a specific profile for all packets entering the router via the LAN interface and leaving it over the DMZ interface.

Routing

Once a packet traversed all ingress packet filters (controlled by the attached profiles), the router decides whether the packet is destined to an application of the gateway itself or shall be routed to a remote host. For this purpose it performs a best-prefix match on the destination IP address in the routing-table, which was previously selected. If no routing-table has been selected explicitly, the *DEFAULT* table is consulted.

If the packet is to be sent to a remote host, it traverses the egress filters of the IP interface (depicted in [figure 34](#)), an egress ACL, another possibility to classify the packet, NAPT translations and finally, a service-policy profile, which can be used to map an internal traffic-class to IP TOS field values.

Packet Processing To/From Local Applications

If the packet is not sent to a remote host, and is destined for a local application (e.g. CLI, the web server, or SIP signaling packets), another set of packet-processing filters is traversed after the routing decision has been made. In particular, another ACL profile dedicated only for locally-terminated flows is passed. This allows you to create specific ACL profiles to protect the local device while having different ACL profiles for routed traffic.

After passing the ACL, voice data packets (RTP/SRTP) are diverted to the voice processing engine whereas the remaining traffic reaches one of the running service applications.

Packets that have been generated by applications on the device also traverse a set of packet-processing filters—a classifier to tag packets with a traffic-class, routing-table selection, and another outbound ACL for locally-generated traffic.

As shown at the top of [figure 34](#) on page 283, the local packet-processing filters are not attached to a specific logical IP interface. All packets to/from a local application rather pass the same set of filters. There is a special local mode within the IP context in which classifier and ACL profiles for local applications can be attached (see chapter 39, “[Classifier Configuration](#)” on page 450 and chapter 38, “[Access Control List Configuration](#)” on page 440 for more information). The local mode also hosts routing-selection commands for locally-generated traffic (see chapter 23, “[IP Routing](#)” on page 300 for more information).

IP Context Overview Configuration Task List

As previously described, this chapter outlines the IP context configuration. It does not give you all the details of a configuration task, but refers you to the chapters in which you will find the full description.

- To view additional information for configuring an IP Interface, refer to chapter 22, “[IP Interface Configuration](#)” on page 289
- To configure a physical port and bind it to a logical IP interface, refer to chapter 14, “[Ethernet Port Configuration](#)” on page 197
- For information on configuring the classifier to tag packets with a traffic-class, refer to chapter 39, “[Classifier Configuration](#)” on page 450
- For information regarding network address port translation (NAPT), refer to chapter 30, “[NAT/NAPT Configuration](#)” on page 372
- For information on setting up the IP router contained in Trinity, refer to chapter 23, “[IP Routing](#)” on page 300
- For essential knowledge related to network security requirements, refer to chapter 38, “[Access Control List Configuration](#)” on page 440
- If your network provides better service to selected network traffic, chapter 40, “[Service Policy Configuration](#)” on page 458 will help you to get in-depth knowledge about quality of service (QoS) management.

The following sections describe the basic tasks involved in IP context configuration. Many parameters have acceptable default values, which in most cases do not need to be explicitly configured. Hence not all of the configuration tasks below are required. Depending on your application scenario, some tasks are mandatory while others are optional. The following tasks use a bottom-up approach, starting from the ports, followed by the

interfaces and up to the services running on the device. Read through the tasks in order to learn a general understanding of the whole network before moving onto more detailed instructions.

- Planning your IP configuration (see [page 286](#))
- Configuring physical ports (see [page 286](#))
- Creating and configuring IP interfaces (see [page 287](#))
- Configuring packet classification (see [page 287](#))
- Configuring Network Address Port Translation (NAPT) (see [page 287](#))
- Configuring static IP routing (see [page 287](#))
- Configuring Access Control Lists (ACL) (see [page 288](#))
- Configuring quality of service (see [page 288](#))

Planning Your IP Configuration

The following subsections provide network connection considerations for Ethernet ports. Patton recommends that you draw a network overview diagram displaying all neighboring IP devices. Do not begin configuring the IP context until you have completed the planning of your IP environment.

IP Interface Related Information

Setting up the basic IP connectivity for your device requires at least the following information:

- IP addresses used for Ethernet LAN and WAN ports
- IP Subnet mask used for Ethernet LAN and WAN ports
- Length for Ethernet cables
- IP addresses of the central SIP registrar
- IP addresses of the central PSTN gateway for SIP-based calls

QoS Related Information

Check with your access service provider if there are any QoS-related requirements, which you need to know prior to configuring Trinity QoS management. Check the following with your access service provider:

- What is the dedicated bandwidth, which you have agreed with your access service provider?
- How does your provider perform packet classification, e.g. which ToS bits have to be used to define the supported classes of service?

Configuring Physical Ports

Port configuration includes parameters for the physical and data link layer, such as framing and encapsulation formats or media access control. Before any higher-layer user data can flow through a physical port, you must associate that port with an interface within the IP context. This association is referred to as a binding. For information and examples on how to configure ports, refer to the respective port type's chapter.

Creating and Configuring IP Interfaces

The number and names of IP interfaces depend upon your application scenario. An interface is a logical construct that encapsulates network-layer protocol and service information, such as IP addressing. Therefore, interfaces are configured as part of the IP context (the virtual router) and represent logical entities that are only usable if a physical port (Ethernet) or circuit (VLAN) is bound to them.

An interface name can be any arbitrary string, but for ease of identification use self-explanatory upper-case names that describe the use of the interface, e.g. LAN, WAN.

Several IP-related configuration parameters are necessary to define the behavior of such an interface. The most obvious parameters are one or multiple IP addresses and the IP net masks that belong to them. Several profile types can also be attached to an IP interface to define how packets arriving on the interface or leaving over it are processed.

For information and examples on how to create and configure an IP interface, refer to [#<IP interface configuration>#](#). The configuration of each profile type is described in a dedicated chapter, and is briefly introduced below.

Configuring Packet Classification

A classifier profile can be attached to each IP interface. It contains rules to match packet flows based on the header fields of the packets and tag them with an internal traffic-class. This traffic-class is usually used in conjunction with other services. For example, an ACL may have filter rules that drop all packets tagged with a certain traffic-class, or policy routing may be configured to select a dedicated routing-table for a packet flow of a given traffic-class. Trinity tests packets against the classifier rules one by one. The first match determines the traffic-class. Because Trinity stops testing rules after the first match, the order of the classifier rules is critical. If no conditions match or if there is no classifier profile attached to an interface, the software tags receive packets with the DEFAULT traffic-class, whereas all packets generated by local applications are tagged with the LOCAL-DEFAULT traffic-class, except generated RTP/SRTP packets, which are tagged as LOCAL-VOICE.

Classifier profiles can be attached to several entities in Trinity-on any local IP interface and in the local mode of the IP context. In both places classifier profiles can be attached separately for inbound and outbound packets.

For an in-depth elaboration of how to configure and use classifier profiles, refer to chapter 39, “Classifier Configuration” on page 450. To get a detailed understanding of how to build packet matching rules, consult chapter 40, “Packet Matching” on page 458.

Configuring Network Address Port Translation (NAPT)

You can configure NAPT by creating a profile that is afterwards used on an explicit IP interface. In Trinity terminology, an IP interface uses a NAPT profile, as shown in [figure 33](#) on page 281. For information and examples on how to configure NAPT, refer to chapter 30, “NAT/NAPT Configuration” on page 372.

Configuring Static IP Routing

Trinity allows you to define static routing entries, which are destination-address-to-egress-interface mappings established by the network administrator prior to the beginning of routing. These mappings do not change unless the network administrator alerts them. Algorithms that use static routes are simple to design, and work well in environments in which network traffic is relatively predictable and where network design is relatively simple.

Routing entries are grouped in routing-tables. A set of route commands in the IP interface can be used to select the routing-table for inbound traffic for different packet-header fields. The route command in the local mode,

within the IP context configures the routing-table to consult for locally-generated traffic. Trinity tests packets against the routing-table-selection rules one by one. The first match determines the routing-table to use. Because Trinity stops testing rules after the first match, the order of the routing-selection rules is critical. If no conditions match or if there is no route command in the interface, the software uses the DEFAULT routing table.

For information and examples on how to configure static IP routing, refer to #< IP routing configuration>#. To get a detailed understanding of how to build packet matching rules, consult #<Packet matcher overview>#.

Configuring Access Control Lists (ACL)

Packet filtering helps to control packet movement through the network. Such control can help to limit network traffic and restrict network use by certain users or devices. An access control list is a sequential collection of permit and deny conditions that apply to packets on a certain interface. Access control lists can be configured for all routed network protocols (IP, ICMP, TCP, UDP, and SCTP) to filter the packets of those protocols as the packets pass through a device. Trinity tests packets against the conditions in an access list one by one. The first match determines whether Trinity accepts or rejects the packet. Because Trinity stops testing conditions after the first match, the order of the conditions is critical. If no conditions match, the software rejects the address.

For information and examples on how to configure access control lists, refer to chapter 38, “[Access Control List Configuration](#)” on page 440. To get a detailed understanding of how to build packet matching rules, consult chapter 40, “[Packet Matching](#)” on page 458.

Configuring Quality of Service (QoS)

A service-policy profile can be attached to an IP interface to manage QoS for network traffic, as shown in [Figure 33](#) on page 281. QoS refers to the ability of a network to provide improved service to selected network traffic over various underlying technologies including Ethernet and 802.x type networks, as well as IP-routed networks. In particular, QoS features provide improved and more predictable network service by providing the following features:

- Supporting dedicated bandwidth
- Improving loss characteristics
- Avoiding and managing network congestion
- Shaping network traffic
- Setting traffic priorities across the network

The QoS features described in chapter 40, “[Service Policy Configuration](#)” on page 458 address these diverse and common needs.

Chapter 22 IP Interface Configuration

Chapter contents

Introduction	290
IP Interface Configuration Task List	290
Creating an IP Interface	290
Deleting an IP Interface	291
Setting the Static IP Address and Network Mask	292
Deleting an IP Address	293
Displaying IP Interface Information	294
Displaying Dynamic ARP Entries	296
ARP Garbage-Collector	296
Gratuitous ARP	296
Testing Connections with the Ping Command	297
Traceroute Command	297
Debugging the IP Configuration	298

Introduction

This chapter provides a general overview of IP interfaces and describes the tasks involved in their configuration. An interface is a logical entity that provides higher-layer protocol and service information, such as Layer 3 addressing. Interfaces are configured as part of a context and are independent of physical ports and circuits.

The separation of the interface from the physical layer allows for many advanced features. For higher layer protocols to become active, a physical port or circuit must be bound to an interface. IP interfaces can be bound physically to Ethernet or DSL ports, or to VLANs, according to the appropriate transport network layer.

IP Interface Configuration Task List

To configure interfaces, perform the tasks in the following sections:

- Creating an IP interface (see [page 290](#))
- Deleting an IP interface (see [page 291](#))
- Setting the static IP address (see [page 292](#))
- Deleting an IP address (see [page 292](#))
- Requesting an IP address via DHCP (see chapter 31, “DHCP Configuration” on page 380)
- Displaying IP interface information (see [page 293](#))
- Testing connections with the **ping** command (see [page 294](#))
- Traceroute command (see [page 297](#))
- Debugging the IP configuration (see [page 298](#))

Creating an IP Interface

Interface names can be any arbitrary string. Use self-explanatory names for your interfaces, which reflect their usage, e.g. LAN, WAN, DMZ.

Mode: configure

Step	Command	Purpose
1	device(cfg)#context ip [ROUTER]	Enters the IP context configuration mode for the default virtual router.
2	device(ctx-ip)[ROUTER]#interface name	Creates the new interface <i>name</i> , which represents an IP interface. This command also places you in interface configuration mode for the interface just created.
3	device(if-ip)[ROUTER.name]#	You are now in the interface configuration mode, where you can enter specific configuration parameters for the IP interface <i>name</i> , e.g., create an IP address.

Example: Create IP interface

The procedure illustrated below assumes that you would like to create an IP interface named *WAN*. Use the following commands in *operator exec* mode.

```
device>enable
```

```
device#configure
device(cfg)#context ip ROUTER
device(ctx-ip)[ROUTER]#interface WAN
device(if-ip)[ROUTER.WAN]#
```

Deleting an IP Interface

Almost every configuration command has a **no** form. In general, use the **no** form to disable a feature or function. Use the command without the **no** keyword to re-enable a disabled feature or to enable a feature that is disabled by default.

Deleting an existing interface in the IP context is often necessary. You can only delete an IP interface if it is not bound from a physical port or circuit. Go to the configuration mode of the physical port or circuit and unbind the interface first before deleting it.

Mode: configure

Step	Command	Purpose
1	device(cfg)#context ip [ROUTER]	Enters the IP context configuration mode for the default virtual router.
2	device(ctx-ip)[ROUTER]#no interface name	Deletes the existing interfaces <i>name</i> .

Example: Delete IP interface

The procedure below assumes that you would like to delete an IP interface named *WAN*, which is currently bound by Ethernet port 0/1. Use the following commands in *operator exec* mode to unbind and delete the IP interface.

```
device>enable
device#configure
```

Unbind port Ethernet 0/1 from the IP interface WAN:

```
device(cfg)#port ethernet 0 1
device(prt-eth)[0/1]#no bind interface
device(prt-eth)[0/1]#exit
```

List the existing interfaces:

```
device(cfg)#context ip ROUTER
device(ctx-ip)[ROUTER]#no interface <?>
LAN Existing IP interface: LAN
WAN Existing IP interface: WAN
```

Delete the IP interface named *WAN* with the no interface command:

```
device(ctx-ip)[ROUTER]#no interface WAN
```

List the interfaces again to check if the appropriate interface was deleted:

```
device(ctx-ip)[ROUTER]#interface <?>
<interface>New IP interface
LAN Existing IP interface: LAN
```

Setting the Static IP Address and Network Mask

Each IP interface needs at least one IP address and an appropriate network mask to be operational. You can use the `ipaddress` interface configuration command to create, change or delete an IP address.

An IP interface may host more than one IP address. Each IP address is, therefore identified by a unique *label*, i.e., a non-empty string. This label must be specified as a parameter to `ipaddress` when creating, changing or deleting the IP address. The same label may be reused in a different IP interface.

An IP address must be unique within the same IP context. That is, you cannot configure the same IP address for the same or different IP interfaces belonging to the same virtual router. It is, however, possible to configure multiple IP addresses within the same subnet, e.g. 10.1.1.1/24 and 10.1.1.2/24.

To learn how to obtain a dynamic IP address via DHCP, see chapter 31, “DHCP Configuration” on page 380.

The `ipaddress` command offers the following options:

Parameter	Explanation
label	Name of the IP address. An IP interface may host more than one IP address. The label identifies the address when changing or deleting it. The label must be unique within the IP interface; you may reuse the same label in a different IP interface. Note The label parameter is optional. If you don't provide it, Trinity automatically generates a default label, which is identical to the name of the IP interface. If you specify <i>DHCP</i> as label, you create a DHCP address (see chapter 31, “DHCP Configuration” on page 380).
address	The network address and mask size in dotted-decimal format a.b.c.d/m for IPv4 or in the colon format a:b:c::x/m for IPv6. Alternatively, you may enter the network address and full network mask with two consecutive parameters, a.b.c.d e.f.g.h or a:b:c::x e:f:g::y, respectively.

Mode: configure

Step	Command	Purpose
1	<code>device(cfg)#context ip [ROUTER]</code>	Enters the IP context configuration mode for the default virtual router.
2	<code>device(ctx-ip)[ROUTER]#interface name</code>	Enters the configuration mode of an existing IP interface or creates a new IP interface.
3	<code>device(ip-if)[ROUTER.name]#ipaddress label address</code>	Creates a new IP address and network mask on the IP interface <i>name</i> or changes the IP address and network mask of an existing IP address (identified by its label).

Example: Create a new static IP address

To create an IP address to 192.168.1.3 with network mask to 255.255.255.0 and label *MAIN* on the IP interface *LAN*, use the following commands in *administrator exec* mode.

```
device>enable
device#configure
device(cfg)#context ip ROUTER
device(ctx-ip)[ROUTER]#interface LAN
```



```
device(ip-if)[ROUTER.LAN]#ipaddress MAIN 192.168.1.3 255.255.255.0
```

Example: Modify an existing static IP address

List the IP addresses within the current interface:

```
device(ip-if)[ROUTER.LAN]#ipaddress <?>
<label>    Unique label of the address within the interface
MAIN      Existing IP address: 192.168.1.3/24
ALT       Existing IP address: 10.0.0.1/8
```

Change the IP address MAIN to 192.168.1.4 and give it a network mask of 255.255.255.0, or a mask size of 24, respectively:

```
device(ip-if)[ROUTER.LAN]#ipaddress MAIN 192.168.1.4/24
```

Deleting an IP Address

Since an IP interface host multiple IP address, Trinity supports to delete them with the no ipaddress command, specifying the address label as the only argument. After the IP address has been deleted, the device is no longer reachable over that address.

Mode: configure

Step	Command	Purpose
1	device(cfg)#context ip [ROUTER]	Enters the IP context configuration mode for the default virtual router.
2	device(ctx-ip)[ROUTER]#interface name	Enters the configuration mode of an existing IP interface or creates a new IP interface.
3	device(ip-if)[ROUTER.name]#no ipaddress label	Deletes an existing IP address.

Example: Delete an existing IP address

To delete the IP address with label *MAIN* on the IP interface *LAN*, follow the procedure below, starting in *administrator exec* mode.

```
device>enable
device#configure
device(cfg)#context ip ROUTER
device(ctx-ip)[ROUTER]#interface LAN
```

List the IP addresses within the current interface:

```
device(ip-if)[ROUTER.LAN]#ipaddress <?>
<label>    Unique label of the address within the interface
MAIN      Existing IP address: 192.168.1.3/24
ALT       Existing IP address: 10.0.0.1/8
```

Delete the IP address with label MAIN:

```
device(ip-if)[ROUTER.LAN]#no ipaddress MAIN
```

List the IP addresses again to check if the appropriate address was detected.

```
device(ip-if)[ROUTER.LAN]#ipaddress <?>
<label>    Unique label of the address within the interface
ALT        Existing IP address: 10.0.0.1/8
```

Displaying IP Interface Information

The show ip interface command displays a list of all IP interfaces and their configured IP addresses. The command is available in operator exec mode or in any of its sub-modes (administrator exec or any configuration mode). By specifying the context and/or IP-interface parameter you can selectively display information for a single context/interface.

Parameter	Explanation
context	Name of the IP context. If the user neither specifies a context nor an interface, the command lists all interfaces in all contexts. If the user specifies a context but no interface, the command lists all interfaces in the specified context.
interface	Name of the IP interface. If the user specifies an interface, information for all interfaces in the specified or default (ROUTER) context is displayed.
detail	Level of detail of the output. If not specified, minimal information is printed in a compact form. The detail levels 2 and 3 provide more information about the status of the IP interface(s) addresses.
full-detail	Prints the maximum amount of information for the IP interface(s) and addresses (=detail level 3)
continuously	Prints an update of the information every second. Press <ctrl><c> to abort the command.

Mode: any

Step	Command	Purpose
1	device>show ip interface [context] [interface] [detail detail full-detail] [continuously]	Displays information about the configuration and state of the selected IP interface(s) and addresses.

Example: Displaying compact information for all IP interfaces

The following example shows how to display information for all IP interfaces by using the show ipinterface command from *operator exec* mode.

```
device>show ip interface
IP Context: ROUTER
=====
IP Interface  Status  Address  Type    Status  Configured  Operational
-----
LAN           up      LAN      static  up      10.1.1.1/24  10.1.1.1/24
              up      DHCP     DHCP    up      (none)      172.168.1.10/24
WAN           up      WAN      static  up      10.1.2.1/24  10.1.2.1/24
```

The columns have the following meaning:

Column	Explanation
IP interface	Name of the IP interface
Status	State of the IP interface: <ul style="list-style-type: none"> • unbound: No physical port or circuit is bound to the IP interface • shutdown: The bound physical port or circuit is administratively shut down; use the <i>no shutdown</i> command on the port/circuit to bring it up. • down: The link of the bound physical port or circuit is down • up: The link of the bound physical port or circuit is up; the IP interface is ready to send/receive packets.
Address	Label of an IP address on the IP interface
Type	IP address type (static or DHCP)
Status	State of the IP address: <ul style="list-style-type: none"> • down: The IP interface is not ready, i.e., not in the up state • FAILED: Requesting a dynamic address or applying a static address failed; execute the command <i>show ip interface full-detail</i> to display the reason of the problem • released: The DHCP address is waiting for a new lease from the DHCP server • applying: The IP address is being applied to the system • revoking: The IP address is being removed from the system, e.g., prior to applying a newly configured address. • up: The device is reachable via the IP address.
Configured	The configured static IP address or requested DHCP address
Operational	The active IP address of the device

Example: Displaying detailed information for a specific IP interface.

The following example shows how to display detailed information for a specific IP interfaces by using the show ipinterface command from operator exec mode:

```
device>show ip interface ROUTER LAN full-detail
IP Interface: LAN
=====
Status:                               up
Port/Circuit:                          ethernet 0 0 (dev: eth0)
Active/Total IPv4 Addresses:           2/2

Address: LAN
-----
Type:                                   static
Status:                                 up
Configured:                             172.16.46.10/19
Operational:                             172.16.46.10/19

Address: DHCP
```

```

-----
Type:                DHCP
Status:              up
Configured:          (none)
Operational:         172.16.60.4/19

```

In addition to the compact form, the detailed version shows the bound physical port or circuit as well as the active (state=up) and total number of IP addresses on the interface.

Displaying Dynamic ARP Entries

The following command can be used to display both the statically-configured as well as the dynamically-learned ARP entries of the device.

Mode: any

Step	Command	Purpose
1	device>show arp	Displays the ARP entries of the system

ARP Garbage-Collector

The following command enables a garbage-collector that will remove ARP-table entries in a STALE state as soon as possible.

Mode: configure

Step	Command	Purpose
1	node(cfg)# arp garbage-collect	Enable the aggressive removal of ARP-table entries that are marked STALE

Gratuitous ARP

The following CLI commands have been added to enable sending and receiving of gratuitous ARP packets. These commands are especially useful for backup nodes that share the same IP address with different MAC addresses. When a node's IP address comes up, it will notify any neighbors of its IP/MAC address so they can update their ARP tables. Similarly, nodes that are configured to receive gratuitous ARP packets will accept these packets and update their local ARP table. Use caution when allowing a device to accept these packets however, as any gratuitous ARP packets received will change the local ARP table..

Mode: configure

Step	Command	Purpose
1	node(cfg)# arp gratuitous emit	Emit gratuitous ARP packets to update neighboring node ARP tables.

Mode: configure

Step	Command	Purpose
1	node(node(cfg)# arp gratuitous accept	Allow received gratuitous ARP packets to update the local ARP table.

Testing Connections with the Ping Command

As an aide to diagnosing basic network connectivity, many network protocols support an echo protocol. The protocol involves sending a special ICMP datagram to the destination host, then waiting for a reply datagram from that host. Results from this echo protocol can help in evaluating the path-to-host reliability, delays over the path and whether the host can be accessed or is functioning.

Mode: any

Step	Command	Purpose
1	device>ping address [count]	Sends <i>count</i> ICMP ECHO_REQUEST packets to the network host at IP address <i>address</i> . If the <i>count</i> parameter is not specified, five requests are sent. The user may abort the command by pressing <ctrl><c>.

When using **ping** for fault isolation, you should first run it on the respective local IP interface to verify that the local LAN or WAN interface is up and running. Then, you should “ping” hosts and gateways farther away. The command computes round-trip times and packet loss statistics when it terminates. If duplicate packets are received, they are not included in the packet loss calculation, although the round-trip time of these packets is used to calculate the minimum/average/maximum round-trip time numbers. When the command terminates, a brief summary of the calculation is displayed.

Example: Testing connections with the **ping** command

The following example shows how to invoke the echo protocol to the destination host at IP address 172.16.1.10 by using the **ping** command from *operator exec* mode.

```
device>ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=45 time=15.019 ms
64 bytes from 8.8.8.8: seq=1 ttl=45 time=52.068 ms
64 bytes from 8.8.8.8: seq=2 ttl=45 time=83.353 ms
64 bytes from 8.8.8.8: seq=3 ttl=45 time=92.690 ms
64 bytes from 8.8.8.8: seq=4 ttl=45 time=32.056 ms
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 15.019/55.037/92.690 ms
```

Traceroute Command

The traceroute command allows you to determine the path a packet takes in order to get to a destination from a given source by returning the sequence of hops the packet has traversed.

Mode: operator

Step	Command	Purpose
1	<code>>tracertoute <host> [destination-port <port>] [max-ttl <hops>] [first-ttl <hops>] [packet-size <bytes>] [probe-count <probes>] [source-address <ipaddress>] [timeout <seconds>] [verbose]</code>	Find the path a packet with a size of bytes takes to a host. Tracertoute will list all hops after first-ttl until max-ttl and it will stop if a hop does not answer after timeout. For each probe a packet will be sent.

Debugging the IP Configuration

When a configured IP address cannot be applied to the system, use the `show ip interface full-detail` command to inspect the reason. The `debug ip` command may also be helpful to obtain a trace of all internal actions while the IP interface is re-configured. In addition, the `debug dynif` command traces link state changes of physical port and circuits.

Mode: any

Step	Command	Purpose
1	<code>device>[no] debug ip</code>	Enables or disables the IP debug monitor
2	<code>device>[no] debug dynif</code>	Enables or disables the link state debug monitor

Example: Debug output while reconfiguring an IP address

The following example switches on the IP debug monitor and then changes the IP address with label WAN on IP interface WAN from 10.1.1.1/24 to 10.1.1.2/24.

```

device>debug ip
device>configure
device#context ip ROUTER
device(ctx-ip)[ROUTER]#interface WAN
device(ip-if)[ROUTER.WAN]#ipaddress WAN 20.1.1.2/24
04:40:24 IP # [DBG] [ROUTER.WAN.WAN] onUpdate: cfgAddress
04:40:24 IP # [INF] [ROUTER.WAN.WAN] State: up (normal) | Event: configuration
update
04:40:24 IP # [INF] [ROUTER.WAN.WAN] State: up (normal) | Action: revoke address
20.1.1.1/24 from dev eth1.1
04:40:24 IP # [INF] [eth1.1] Kernel << ip addr del 20.1.1.1/24 dev eth1.1
04:40:24 IP # [INF] [eth1.1] Kernel >> Address departed: WAN (20.1.1.1/24)
04:40:24 IP # [INF] [ROUTER.WAN.WAN] State: up (normal) -> revoking (reconfigure)
04:40:24 IP # [INF] [ROUTER.WAN.WAN] State: up (normal) | Action: decrement
active

address counter
04:40:24 IP # [DBG] [ROUTER.WAN] onUpdate: activeV4AddrCount
04:40:24 IP # [INF] [ROUTER.WAN.WAN] State: revoking (reconfigure) | Event: down
04:40:24 IP # [INF] [ROUTER.WAN.WAN] State: revoking (reconfigure) | Action:
restart

(reconfigure)
04:40:24 IP # [INF] [ROUTER.WAN.WAN] Restart Mode: reconfigure -> normal
04:40:24 IP # [INF] [ROUTER.WAN.WAN] State: revoking (normal) | Action: apply
address

```

```
                20.1.1.2/24 to dev eth1.1
04:40:24 IP # [INF] [eth1.1] Kernel << ip addr add 20.1.1.2/24 broadcast + label
                eth1.1:WAN dev eth1.1
04:40:24 IP # [INF] [ROUTER.WAN.WAN] State: revoking (normal) -> applying
04:40:24 IP # [DBG] [ROUTER.WAN.WAN] onUpdate: address, restartMode, state
04:40:24 IP # [INF] [eth1.1] Kernel >> Address arrived: WAN (20.1.1.2/24)
04:40:24 IP # [INF] [ROUTER.WAN.WAN] State: applying (normal) | Event: up
04:40:24 IP # [INF] [ROUTER.WAN.WAN] State: applying (normal) -> up
04:40:24 IP # [INF] [ROUTER.WAN.WAN] State: applying (normal) | Action: increment
active
                address counter
04:40:24 IP # [DBG] [ROUTER.WAN.WAN] onUpdate: state
04:40:24 IP # [DBG] [ROUTER.WAN] onUpdate: activeV4AddrCount
```

Chapter 23 IP Routing

Chapter contents

Introduction	301
Basic Routing	301
Static Routes	301
Configuring static routes	301
System Routes	302
Dynamic Routes	302
Show Routes	303
Basic Static Routing Example	303
Policy Routing.....	304
Routing Tables	305
Creating a table	306
Configuring static routes	306
Show routes	306
Traffic Assignment	306
Assign an IP Interface	307
Assignment by Rules	308
Assignment by Traffic-Class	310

Introduction

The Trinity IP Routing facility consists of the two major functionalities *Basic Routing* and *Policy Routing*.

Basic Routing

Under Basic Routing is to be understood the destination IP address based next-hop determination. The next-hop or gateway selection is done by matching a set of routing rules entered by the user (static-route), received through a routing-protocol (dynamic-route) or added by the system (system-route). Routing entries which specify a *gateway* as next-hop are also called **gateway-routes**. Networks that are directly reachable through a device's network-port are specified through **interface-routes**. Instead of a gateway they specify an outgoing interface.

In the *context ip* configuration mode exists a system pre-created routing-table called *DEFAULT*. This table contains all *Basic Routing* information and cannot be deleted by the user. Actually it is possible to create additional routing-tables with a user defined name but such user-created tables are part of the *Policy Routing* and do not have any use in *Basic Routing*.

All Basic Routing features are available for IPv4 as well as for IPv6.

Static Routes

These are user managed gateway and interface routes and are getting exported in the running-config. In the output of the *show route* command they are flagged with an '**R**'. Another flag '**U**' indicates if the route is **up** or not. A static gateway-route is becoming active (up) if the gateway is reachable. For this we need the following conditions:

- At least one IP address in the gateway's network has to be configured.
- The IP interface which owns the IP address has to be bound from a network-port.
- The network-port's link state has to be **up**.

A static interface-route is becoming active (up) if:

- The specified outgoing interface is bound from a network-port.
- The specified outgoing interface has at least one IP address configured.
- The network-port's link state has to be **up**.

Configuring static routes

A route is clearly identified by its destination address/mask combination and the metric. That means it is allowed to configure several time the same destination, using the same or different gateways, but with a different metric value. The metric in a static route has the meaning of a priority where lower value means higher priority.

Static route differentiation by metric is useful if a destination network is reachable through different gateways. Usually gateways are located in the same network as the device itself. If the link to the gateway with lowest metric is going down, this static-route is becoming unavailable. In that case the device's router will select the route to the destination with the next higher metric and another gateway is going to be used.

Mode: configure

Step	Command	Purpose
1	<code>[device](cfg)#context ip [ROUTER]</code>	Enters the context IP ROUTER configuration mode.
2	<code>[node](ctx-ip)[ROUTER]#routing-table [DEFAULT]</code>	Enters the routing-table DEFAULT configuration mode.
3	<code>[node][ROUTER.DEFAULT]#route { <network>/<mask-size> <network> <mask> default default-v6 } { gateway <gw-address> interface <if-name> } [metric <metric>]</code> OR <code>[node][ROUTER.DEFAULT]#no route <network>/<mask-size> [metric <metric>]</code>	Adds a static route. Removes a static route.

Syntax:

Parameter	Explanation
network	The destination network address in the dot-format a.b.c.d for IPv4 and in the colon-format a:b:c::x for IPv6.
mask-size	Number of mask-bits defining the destination network.
mask	The destination network mask in the dot-format a.b.c.d for IPv4 and in the colon-format a:b:c::x for IPv6.
default	Short form for defining a default IPv4 route. It configures network/mask-size with 0.0.0.0/0 .
default-v6	Short form for defining a default IPv6 route. It configures network/mask-size with ::/0 .
gw-address	The address of the next-hop router that can access the destination network. In the dot-format a.b.c.d for IPv4 and in the colon-format a:b:c::x for IPv6.
interface	The name of the outgoing interface to be used for reaching the destination network.
metric	Metric value of the route. Default: 0

System Routes

For each assigned IP address the system automatically creates route entries for the belonging network into the *DEFAULT* routing-table. That means, all directly available networks are known by the system and don't have to be configured. The system-routes are of type *interface-route* means, only the outgoing interface is specified and do not have the gateway parameter. In the output of the show route command they are flagged with an 'S'.

Dynamic Routes

This kind of routes is assigned to the system either by a routing-protocol (RIP, BGP ...) or through a device configuration protocol (DHCP, PPP ...). In the output of the *show route* command they are flagged with a 'D'. A *dynamic-route* is active under the same condition as a *static-route*.

Show Routes

Execution of *show running-config* command only displays the static-routes which have been added to the system. Neither dynamic-routes nor system-routes are shown there. To get an overview of all routes actually known by the system the *show route* command has to be executed.

Mode: Operator execution

Step	Command	Purpose
1	[node]#show route [<detail>]	Displays route information Default: 0

Output:

```

Routing Tables
=====
Flags: C - dhCp, D - Dynamic, G - use Gateway, H - target is a host
       R - useR, U - route is Up, S - System

Routing Table DEFAULT, ID = 254
Destination      Gateway          Flags Metric Interface      Source
172.16.32.0/19   172.16.45.4      SU    0      WAN             172.16.45.7
30.30.30.0/24    172.16.45.4      RU    0
172.16.32.0/19   172.16.45.4      SU    0      WAN             172.16.60.7
0.0.0.0/0        172.16.32.1      CDGU  0
    
```

Basic Static Routing Example

The picture below shows an Internetwork consisting of three routers, a Trinity device in the middle, and the four autonomous networks, with network addresses 10.1.5.0/16, 172.16.40.0/24, 172.17.100.0/24 and 10.2.5.0/16. The Trinity device shall be configured for the following IP routing scenario:

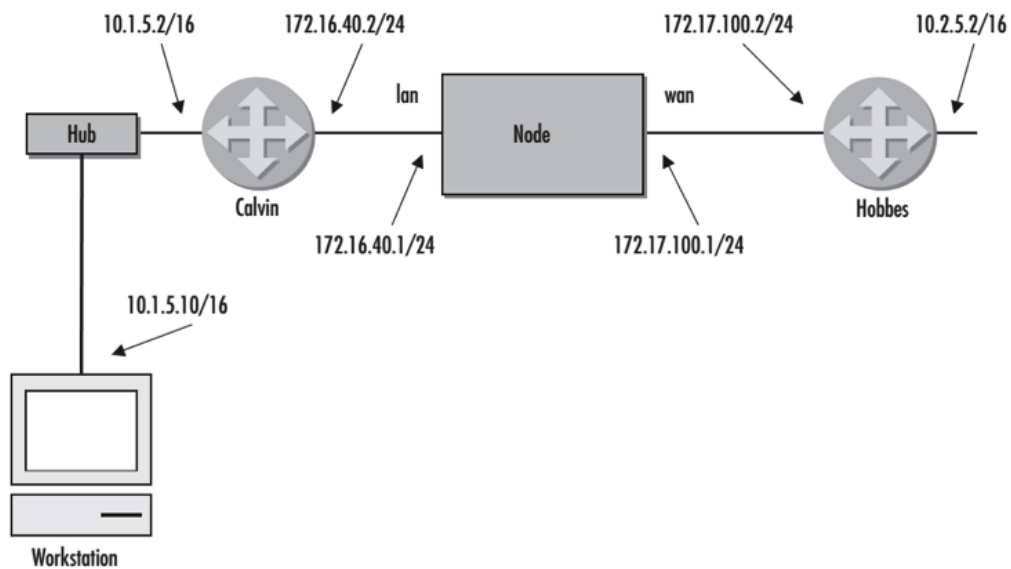


Figure 35. Static route example

All packets for the Workstation with IP address 10.1.5.10 shall be forwarded to the next-hop router Calvin. All packets for network 10.2.5.0/16 shall be forwarded to the next-hop router Hobbes.

Example Configuration:

```
context ip ROUTER

routing-table DEFAULT
  route 10.1.5.10/32 gateway 172.16.40.2 metric 0
  route 10.2.0.0/16 gateway 172.17.100.2 metric 0
```

Show Route Output:

```
Routing Tables
=====
Flags: C - dhCp, D - Dynamic, G - use Gateway, H - target is a host
       R - useR, U - route is Up, S - System

Routing Table DEFAULT, ID = 254
Destination      Gateway          Flags Metric Interface    Source
172.16.40.0/24   172.16.40.2     SU    0     LAN            172.16.40.1
172.17.100.0/24  172.17.100.2    SU    0     WAN            172.17.100.1
10.1.5.10/32     172.16.40.2     RU    0
10.2.0.0/16      172.17.100.2    RU    0
```

Policy Routing

IP routing makes decisions based on IP addresses. Policy Routing allows the user to configure IP routing based on more criteria than only the destination IP address.

The heart of the Trinity policy-routing is the ability of traffic assignment to different routing-tables based on packet-matching at two different observation points. By making use of Trinity's Packet Matcher functionality it is possible to select traffic by more than 20 different criteria in an 'and' conjunction and to assign it to one of up to 251 user defined routing-tables. Several matching rules can be added in a prioritized order.

User defined routing-tables have the same routing features as the system created *DEFAULT* table has. Forwarding is done by destination IP address based next-hop determination (See “[Basic Routing](#)” on page 301). No additional functionality is needed due to assumption the traffic has been separated through packet-matching before it reaches the routing-table.

Observation points for Policy Routing's packet-matching are the IP interfaces and the *local* pseudo interface in context IP.

[Figure 36](#) on page 305 shows an example of a context IP configuration with the Observation Points at which traffic from local applications and from LAN is dispatched to different routing-tables. Each routing-table is then forwarding the packets according to its routing-entries.

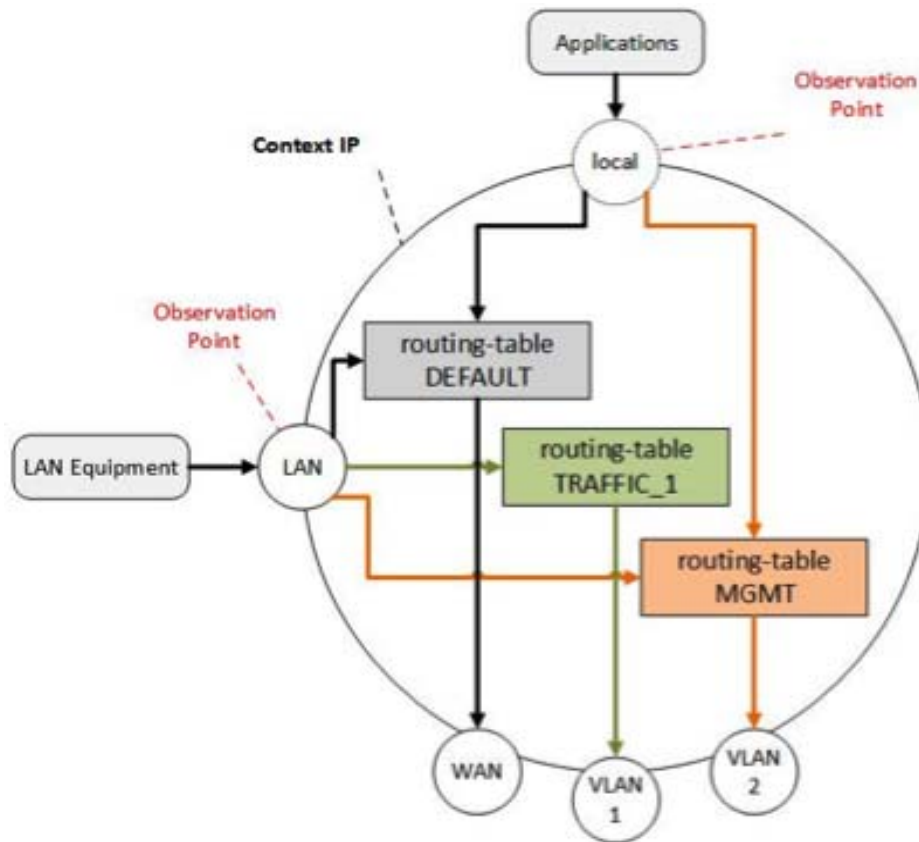


Figure 36. Policy-routing observation points

The different elements in the Policy-Routing Observation Points are:

- local: Pseudo interface in context IP where local application traffic can be matched.
- LAN: IP interface bound to LAN's network port.
- WAN, VLAN1, VLAN2: IP interfaces bound to the WAN's network and VLAN ports.
- DEFAULT: System created routing-table forwards all unspecified traffic.
- TRAFFIC_1, TRAFFIC_2: User-Created routing-tables forwarding a given kind of traffic.

Routing Tables

Besides the system-created *DEFAULT* routing-table it is possible to create up to 251 additional tables. Each user created routing-table has the same possibilities and behavior as the *DEFAULT* table (see also “[Basic Routing](#)” on page 301). The difference is their responsibility. User-Created tables are not involved in the routing process as long as there is not explicit traffic assigned to them (see “[Traffic Assignment](#)” on page 306). On the other hand the *DEFAULT* table is responsible for all traffic that is not explicitly assigned to another routing-table.

On creation of a new routing-table all system-routes (see “[System Routes](#)” on page 302) of all available IP addresses and all dynamic-routes (see “[Dynamic Routes](#)” on page 302) are automatically assigned to the table.

Also upcoming system- and dynamic-routes are replicated to already existing routing-tables. Therefore, user-created tables know the same base networks as the *DEFAULT* table.

Creating a table

Mode: configure

Step	Command	Purpose
1	[device](cfg)#context ip [ROUTER]	Enters the context ip ROUTER configuration mode.
2	[node](ctx-ip)[ROUTER]#routing-table <name> OR [node](ctx-ip)[ROUTER]#no routing-table <name>	Creates a new routing-table if it doesn't yet exist or enters configuration mode if it already exists. Removes a routing-table.

Configuring static routes

Because the static-route configuration syntax is exactly the same as for the *DEFAULT* table, please take a look at Configuring static routes.

Show routes

Because the syntax to display all available routes and the output is the same as for the *DEFAULT* table, please take a look at Show routes.

Traffic Assignment

As mentioned in the introduction, there exists two observation points for assigning traffic to a routing-table. Each IP interface of context IP and the local pseudo interface can be configured to route traffic according to specified criteria to a specific routing table.

All unmatched traffic is automatically sent to the routing-table *DEFAULT*. This is also the default behavior if the user doesn't care about Policy Routing.

In the *interface ip* or the *local* configuration mode explicit rules can be defined to route a specific type of traffic to a routing-table.

Assign an IP Interface

This is the simplest form for assigning traffic to a specific routing-table. All traffic received from an ip interface is sent to a configured routing-table. As visible in [figure 37](#) such a configuration makes sense if the traffic on an interface is just of one category and is accessing always the same domain.

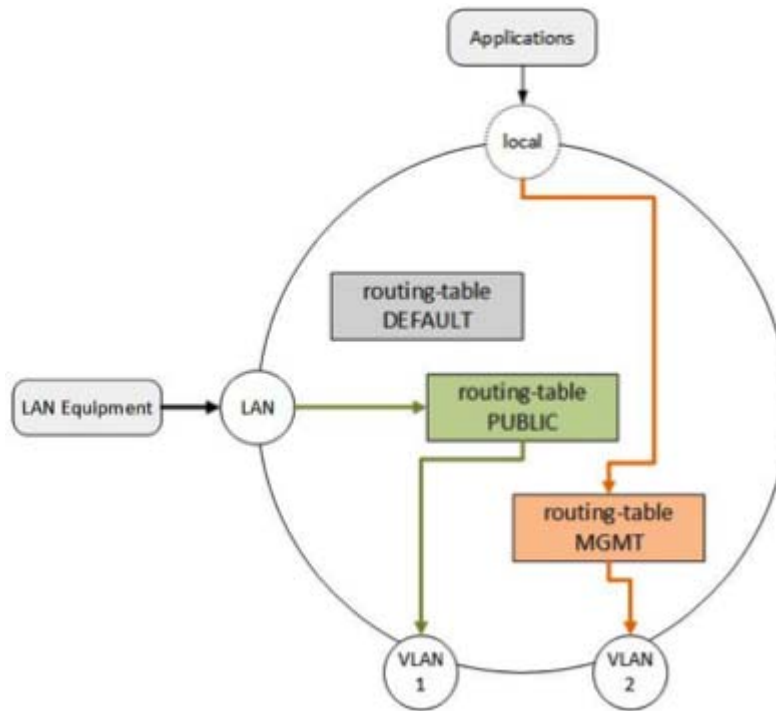


Figure 37. Whole ip interface assignment

Configuration Syntax

Mode: configure

Step	Command	Purpose
1	<code>[device](cfg)#context ip [ROUTER]</code>	Enters the context ip ROUTER configuration mode.
2	<code>[device](ctx-ip)[ROUTER]#interface <if-name></code> OR <code>[device](ctx-ip)[ROUTER]#local</code>	Enters the ip interface configuration mode Enters the local configuration mode.
3	<code>[device](ctx-ip)[ROUTER.name]#route dest-table <table-name></code>	Assigns the whole traffic to a routing-table.

Example Configuration:

```
context ip ROUTER

  local
    route 1 dest-table MGMT

  interface LAN
    ipaddress LAN 10.10.10.1/24
    route 1 dest-table PUBLIC

  interface VLAN1
    ipaddress VLAN1 192.168.100.1/24

  interface VLAN2
    ipaddress VLAN2 192.168.200.1/24

  routing-table DEFAULT

  routing-table PUBLIC
    route 0.0.0.0/0 gateway 192.168.100.2 metric 0

  routing-table MGMT
    route 0.0.0.0/0 gateway 192.168.200.2 metric 0
```

Assignment by Rules

With Trinity's Packet-Matcher functionality it is possible to create user-defined rules to define which traffic has to be sent to which routing-table. The entered rules have a prioritized order where lower order means higher priority. Actually the rules are applied from top to down as they are listed in *show running-config*.

If a packet doesn't match a rule's criteria the next rule is applied to the packet. If a rule matches the packet it is sent to the specified routing-table and the matching process stops. If a packet doesn't match any rule it is forwarded to the *DEFAULT* table.

In [figure 38](#) on page 309 a scenario is shown where traffic is separated into three different categories. All VoIP packets (SIP, RTP) either coming from LAN clients or generated locally have to be handled by routing-table *VOICE*. For managing the device itself a further separation is done through the *MGMT* table. Finally, all traffic from the LAN that doesn't belong to the VoIP category is assumed to access the public internet and is sent to the *PUBLIC* table. Take a look at Example Configuration which is also referring to [figure 38](#).

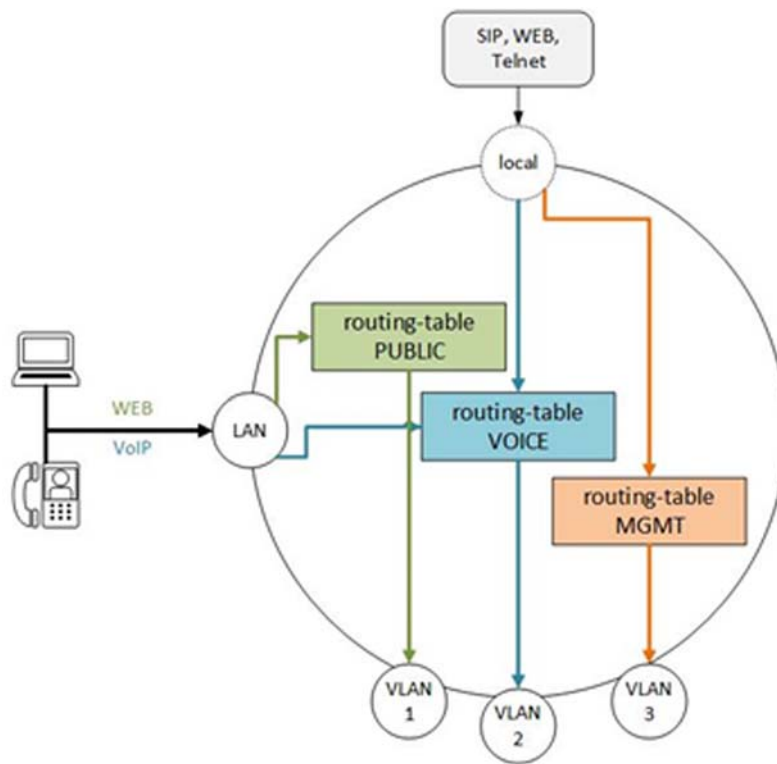


Figure 38. Explicit traffic assignment

Configuration Syntax

For all packet-matcher-options please consult chapter 40, “[Packet Matching](#)” on page 458.

Mode: configure

Step	Command	Purpose
1	<code>[node](cfg)context ip [ROUTER]</code>	Enters the context ip ROUTER configuration mode.
2	<code>[device](ctx-ip)[ROUTER]#interface <if-name></code> OR <code>[device](ctx-ip)[ROUTER]#local</code>	Enters the ip interface configuration mode Enters the local configuration mode.
3	<code>[node](if-ip)[ROUTER.name]#route [<index> { after before <index> }] <packet-matcher-options> dest-table <table-name></code> <code>[node](if-ip)[ROUTER.name]#no route <index></code>	Adds a new routing rule or is moving an existing rule. Deletes the rule at <i>index</i> .

Example Configuration

The configuration below is referring to [figure 38](#) on page 309. In the local configuration SIP signaling originated with UDP or TCP is sent to the *VOICE* table. Local generated RTP traffic is expected from the source

ports 4864 to 5375 and is also sent to table *VOICE*. All traffic from the WEB server (Port 80) and the Telnet server (Port 23) is sent to table *MGMT*.

SIP traffic generated on LAN is expected to be handled the same as local generated SIP traffic. Therefore also SIP signaling (Port 5060) is sent to *VOICE*. On LAN the RTP packets are expected in the UDP source port range of 4000 to 4099. The last rule in the interface LAN configuration forces all traffic that didn't match any rules before to be sent to *PUBLIC* table.

```
context ip ROUTER

local
  route 1 protocol udp src-port 5060 dest-table VOICE
  route 2 protocol tcp src-port 5060 dest-table VOICE
  route 3 protocol udp src-port 4864..5375 dest-table VOICE
  route 4 protocol tcp src-port 80 dest-table MGMT
  route 5 protocol tcp src-port 23 dest-table MGMT

interface LAN
  ipaddress LAN 10.10.10.1/24
  route 1 protocol udp src-port 5060 dest-table VOICE
  route 2 protocol tcp src-port 5060 dest-table VOICE
  route 3 protocol udp src-port 4000..4099 dest-table VOICE
  route 4 dest-table PUBLIC

interface VLAN1
  ipaddress VLAN1 192.168.100.1/24

interface VLAN2
  ipaddress VLAN2 192.168.200.1/24

interface VLAN3
  ipaddress VLAN3 192.168.220.1/24

routing-table DEFAULT

routing-table PUBLIC
  route 0.0.0.0/0 gateway 192.168.100.2 metric 0

routing-table MGMT
  route 0.0.0.0/0 gateway 192.168.200.2 metric 0

routing-table VOICE
  route 0.0.0.0/0 gateway 192.168.220.2 metric 0
```

Assignment by Traffic-Class

In section “Assignment by Rules” on page 308 is explained how to categorize traffic in ip *interface* or in *local* for sending it to a specific routing-table. There the categorization is explicitly done for policy-routing. Because other services probably are also assigned to a specific type of traffic it makes sense to categorize traffic only once and all service can operate on that classification. Trinity provides that feature in framework of chapter 39, “Classifier Configuration” on page 450. With the Classifier it is possible to tag packets with a *traffic-class* in an early stage of the system's packet receive path. Tagging the packets is done through chapter 40, “Packet Matching” on page 458 rules which are configured in a *Classifier Profile*.

The system knows three pre-created and assigned traffic-classes which can directly be used in the IP interface and local route rules.

- local-voice: All locally generated real-time traffic (RTP, SRTP, T.38 etc.)
- local-default: All locally generated application traffic (WEB, SIP, Telnet etc.)
- default: All other traffic (Routed traffic, System messages like ICMP-Redirect etc.)

To enable classifier profile it has to be attached to the interfaces (IP & local) where the packets to be marked are entering or leaving the system. Attaching the profile is done through the use *profile classifier command*. This command has a direction attribute which can be set to *in* or *out*. To use a classifier profile for policy-routing only one direction makes sense. Dependent on the use location it is different:

- IP interface: *in*
- Local: *out*

In [figure 39](#) the classifier directions are shown with the in/out labeled arrows. Because *local* applications are part of the device, their generated traffic is sent *out* and needs to be marked in direction *out* for policy-routing. On the ip interface the traffic that is coming *in* the device has to be marked for policy-routing.

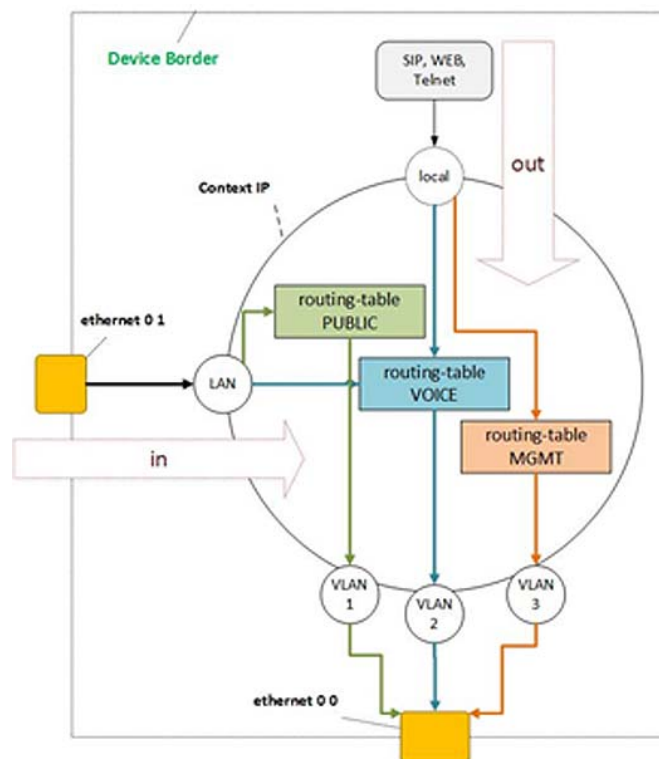


Figure 39. Classifier Directions

Configuration Syntax

For the configuration syntax of a classifier profile, see chapter 39, “[Classifier Configuration](#)” on page 450.

Mode: configure

Step	Command	Purpose
1	<code>[node](cfg)#context ip [ROUTER]</code>	Enters the context ip ROUTER configuration mode.
2	<code>[node](ctx-ip)[ROUTER]#interface <if-name></code> OR <code>[node](ctx-ip)[ROUTER]#local</code>	Enters the ip interface configuration mode. Enters the local configuration mode.
3	<code>[node](if-ip)[ROUTER.name]#use profile classifier { in out } [<index> { after before <index> }] <profile-name></code>	Adds a new classifier-profile or is moving an existing one.
	<code>[node](if-ip)[ROUTER.name]#no use profile classifier { in out } <index></code>	Deletes the classifier-profile at <i>index</i> .

Example Configuration

The following example is referring to scenario [figure 38](#) on page 309. In difference to the previous Example Configuration the same scenario is implemented with the system's known traffic-classes and with classifier profile that is tagging another traffic-class.

One special thing is the processing of the SIP messages. Per default all traffic from local applications is tagged with *local-default*. But because we want to send that kind of traffic to the *VOICE* routing-table it is re-tagged as *sip-sig* through the used classifier profile. The same classifier profile is the re-usable in *local* as well as in LAN interface.

```

profile classifier CL_MGMT
  match 1 protocol tcp src-port 80 set traffic-class TC_MGMT
  match 2 protocol tcp src-port 23 set traffic-class TC_MGMT

profile classifier CL_SIP_SIG
  match 1 protocol udp src-port 5060 set traffic-class TC_SIP_SIG
  match 2 protocol tcp src-port 5060 set traffic-class TC_SIP_SIG

context ip ROUTER

  local
    use profile classifier out 1 CL_SIP_SIG
    use profile classifier out 2 CL_MGMT
    route 1 traffic-class local-voice dest-table VOICE
    route 2 traffic-class TC_SIP_SIG dest-table VOICE
    route 3 traffic-class TC_MGMT dest-table MGMT

  interface LAN
    ipaddress LAN 10.10.10.1/24
    use profile classifier in 1 CL_SIP_SIG
    route 1 traffic-class TC_SIP_SIG dest-table VOICE
    route 2 protocol udp src-port 4000..4099 dest-table VOICE
    route 3 dest-table PUBLIC

  interface VLAN1

```

```
ipaddress VLAN1 192.168.100.1/24

interface VLAN2
  ipaddress VLAN2 192.168.200.1/24

interface VLAN3
  ipaddress VLAN3 192.168.220.1/24

routing-table DEFAULT

routing-table PUBLIC
  route 0.0.0.0/0 gateway 192.168.100.2 metric 0

routing-table MGMT
  route 0.0.0.0/0 gateway 192.168.200.2 metric 0

routing-table VOICE
  route 0.0.0.0/0 gateway 192.168.220.2 metric 0
```

Chapter 24 **Internet Protocol Security (IPsec) Configuration**

Chapter contents

Introduction	316
About IPsec	316
IKE	316
Data transport	316
IPsec configuration task list	317
Install License	317
Configure Time	317
Configure Public-Key Infrastructure (PKI)	317
Configure IPsec profile	318
Creating an IPsec profile and entering configuration mode	319
Entering the Phase 1 configuration mode	319
Configuring the Phase 1	319
Entering the Phase 2 configuration mode	319
Configuring the Phase 2	320
Configure IPsec tunnel	320
Local	320
Remote	320
Local identity	320
Remote identity	320
Bind	320
IPsec profile	320
Authentication method	321
Pre-shared key	321
Public key	321
Match	321
Creating an IPsec tunnel and entering configuration mode	322
Configuring the IPsec tunnel	322
Entering the authentication configuration mode	322
Configuring the authentication	323
Configure traffic selectors	323
Entering configuration mode	324
Configuring the traffic selectors	324
Configure road warrior vs site-to-site VPN	324
Examples	325
IPsec profile configuration	325
IP interface configuration	326
IPsec tunnel configuration	326
Troubleshooting	327

Diffie-Hellman groups and encryption/integrity algorithms	327
Diffie-Hellman groups	327
Encryption algorithms	327
Integrity algorithms	327

Introduction

This chapter gives an overview of the Internet Protocol Security (IPsec), and describes how to use IPsec commands to configure Trinity.

About IPsec

IPsec (see RFC4301) is one of the various technologies available to establish virtual private networks (VPN). A VPN is a private data network that uses the public telecommunications infrastructure, maintaining confidentiality with the use of a tunneling protocol. IPsec is a network protocol suite which provides the following features.

- data origin authentication—Ensures that the peer is who he claims to be
- data integrity—Ensures that data is not tampered with while in transit
- data confidentiality—Protect the data against eavesdropping and unauthorized access

The IPsec network protocol suite includes several protocols:

- IKE
- Data transport

IKE

- The first of two phases of the VPN establishment
- Performs the secure communication establishment of a channel to negotiate other parameters needed for the establishment of child security associations using AH or ESP
- Two versions, IKEv1 and IKEv2

Data transport

- The second of two phases of the VPN establishment
- Only one of the following protocols is used, AH or ESP
- Authentication Header (AH)
 - Performs data integrity only, no encryption
 - Does not work well when a NAT is present
 - AH is almost never used anymore
- Encapsulating Security Protocol (ESP)
 - Performs data integrity and confidentiality
 - Works when a NAT is present
 - ESP is almost always used

The IPsec protocol supports two modes of operation, transport mode and tunnel mode. Trinity is currently limited to the tunnel mode and to being client side only.

IPsec configuration task list

To configure IPsec, perform the tasks in the following list:

- Install license (see page 317)
- Configure time (see page 317)
- Configure Public-Key Infrastructure (PKI) (see page 317)
- Configure IPsec profile (see page 318)
- Configure IPsec tunnel (see page 320)
- Configure traffic selectors (see page 323)
- Configure road warrior vs site to site VPN (see page 324)
- Examples (see page 325)
- Troubleshooting
- Diffie-Hellman groups and encryption/integrity algorithms

Install License

The VPN support is a licensed option for Patton devices. You need to have a *VPN* license installed on your device before you can configure and use IPsec. You may check if the VPN license is installed with the **show system licenses** command.

Configure Time

The system time must be configured before you can work with certificates, and enabling PKI and IPsec. It is also very important to set the correct offset of the local time with respect to the UTC time, due to the validity of certificates expressed in UTC time. The configuration option that works best is to update the UTC time from a configured SNTP server and to derive the local time by configuring the static offset of the local time zone with respect to UTC. If a SNTP server is not available, you should first configure the offset of the local time zone and afterwards set the local time with **clock set** command.

Errors in the form of *THE OWN CERTIFICATE pki:certificate/own-certificate.cert.pem HAS NO ISSUER WITHIN THE OWN CERTIFICATES* when trying to establish an IPsec tunnel are often due to the wrong time being configured.

Note It is important to also correctly configure the time of remote devices, which are accepting connections from our device.

Configure Public-Key Infrastructure (PKI)

The commands that are needed to set up the public-key infrastructure are described in detail in Chapter 35, “Public-Key Infrastructure (PKI)” on page 408. It is responsible for creating, importing and exploring keys and certificates, and storing them persistently in flash memory.

An IPsec tunnel (described in the next section) groups security and other relevant configuration required to establish an IPsec tunnel. For this purpose, the IPsec tunnel may refer to a private key, one or more own-certificates and one or more trusted certificates. To configure and use an IPsec tunnel, you need to create and/or

import keys and certificates as described in Chapter 35, “Public-Key Infrastructure (PKI)” on page 408.

Configure IPsec profile

The IPsec profile is used to configure the algorithms and timeout that will be used by the phase 1 and phase 2 of the IPsec tunnel establishment.

The content of the IPsec profile is generic and can be used for any IPsec tunnel. A newly created IPsec profile is configured with default groups and algorithms with are considered secure. They will often work without any modification. Configuration of the IPsec profile is often optional but depends on the configuration of the IPsec server you wish to establish a tunnel to.

Here is the list of parameters that may be configured for both phases.

- Phase 1
 - Key exchange protocol, ikev1, ikev2 and all for any one of them
 - Diffie-Hellman group
 - Encryption algorithm
 - Integrity algorithm
 - Key lifetime, how long (in seconds) the negotiated key may be used before rekeying
- Phase 2
 - Packet protection protocol, AH or ESP
 - Diffie-Hellman group
 - Encryption algorithm (only for ESP)
 - Integrity algorithm
 - Key lifetime, how long (in kilobytes, in packets or in second) the negotiated key may be used before rekeying.

For the Diffie-Hellman group, encryption and integrity algorithms you may set the value “secure”. It represents a subset of the groups and algorithms that are judged secure. It is expected that the subsets will involve as the technology progresses. Setting an explicit group or algorithm will make sure that the configuration will remain the same between firmware upgrade. You may also set the value “all”. It basically means that all groups and algorithms will be proposed. “secure” and “all” are only supported when the key exchanged protocol “ikev2” is configured. This makes it easy to establish a first tunnel but it is much less secure and should not be used for a production environment.

Creating an IPsec profile and entering configuration mode

This procedure describes how to create an IPsec profile and enter the IPsec profile configuration mode.

Mode: Administrator exec

Step	Command	Purpose
1	<code>node(cfg)# profile ipsec <name></code>	Enter the IPsec profile configuration mode. Mandatory.
	<code>node(pf-ipsec)[name]#</code>	

Entering the Phase 1 configuration mode

This procedure describes how to enter the Phase 1 configuration mode.

Mode: phase-1-ike

Step	Command	Purpose
2	<code>node(pf-ipsec)[name]# phase-1-ike</code>	Enter the IPsec profile configuration mode. Mandatory.
	<code>node(pf-ipsec-phase-1-ike)[name]#</code>	

Configuring the Phase 1

This procedure describes how to configure the Phase 1.

Step	Command	Purpose
3	<code>node(pf-ipsec-phase-1-ike)# key-exchange-protocol <protocol></code>	Set the supported key exchange protocols. Optional. Default is all.
	<code>node(pf-ipsec-phase-1-ike)# diffie-hellman-group <group></code>	Set the Diffie-Hellman group. Optional. Default is modp-2048.
	<code>node(pf-ipsec-phase-1-ike)# encryption-algorithm <algorithm></code>	Set the encryption algorithm. Optional. Default is aes-128.
	<code>node(pf-ipsec-phase-1-ike)# integrity-algorithm <algorithm> seconds</code>	Set the integrity algorithm. Optional. Default is sha2-256.
	<code>node(pf-ipsec-phase-1-ike)# key-lifetime <seconds></code>	Set the key lifetime. Optional. Default is one day.
	<code>exit</code>	

Entering the Phase 2 configuration mode

This procedure describes how to enter the Phase 2 configuration mode.

Mode: phase-2-ipsec

Step	Command	Purpose
4	<code>node(pf-ipsec)[name]# phase-2-ipsec</code>	Enter the phase 2 configuration mode. Optional.
	<code>node(pf-ipsec-phase-2-ipsec)[name]#</code>	

Configuring the Phase 2

This procedure describes how to configure the Phase 2.

Mode: phase-2-ipsec

Step	Command	Purpose
5	<code>node(pf-ipsec-phase-2-ipsec)# packet-protection-protocol <protocol></code>	Set the supported key exchange protocols. Optional. Default is all.
	<code>node(pf-ipsec-phase-2-ipsec)# diffie-hellman-group <group></code>	Set the Diffie-Hellman group. Optional. Default is modp-2048.
	<code>node(pf-ipsec-phase-2-ipsec)# encryption-algorithm <algorithm></code>	Set the encryption algorithm. Optional. Default is aes-128.
	<code>node(pf-ipsec-phase-2-ipsec)# integrity-algorithm <algorithm></code>	Set the integrity algorithm. Optional. Default is sha2-256.
	<code>node(pf-ipsec-phase-2-ipsec)# key-lifetime <seconds> seconds</code>	Set the key lifetime. Optional. Default is one day.
	<code>exit</code>	

Configure IPsec tunnel

The IPsec tunnel represents a secure connection between the client and the server. It is used to configure the parameters that directly pertain to the tunnel establishment.



The configuration of the following parameters is mandatory.

Local

The IP address from which the tunnel will be established.

Remote

The IP address to which the tunnel will be established.

Local identity

Identifies the client. Used by the server during the authentication of the client.

Remote identity

Identifies the server. Used by the client during the authentication of the server.

Bind

The local IP interface that will be associated with the IPsec tunnel.

IPsec profile

The IPsec profile that must be used for the tunnel establishment.

Authentication method

This is the authentication method that will be used to authenticate the server. There are two methods, pre-shared-key (based on a shared secret) and public-key (based on the public key infrastructure. Default method is pre-shared key.

Each authentication method has different parameters. Only the parameters associated with the configured authentication method must be configured.

Pre-shared key

A pre-shared key is a secret key that is known only to both the client and the server.

When the authentication method is pre-shared key the following must also be configured:

- The local identity may be anything but it must be the same on both the client and the server.
- The remote identity may be anything but it must be the same by both the client and the server.

Public key

Based on public key infrastructure (PKI).

When the authentication method is public-key the following must also be configured:

- The certificate chain of the client. It is composed of the client own-certificate, zero or more intermediate certificates and zero or one root certificate. When more than one certificate is present, the chain of trust must be valid. This means that every certificate except the highest one must signed by an authority described by another certificate in the chain. The root certificate is optional because the server needs to already have it in its configuration as a trusted certificate.
- The private key associated with the own-certificate of the client.
- The trusted certificates. You can directly specify one or more specific trusted certificates or you can indicate one of the following subset to simplify the configuration. Default is all-stored.
 - build-in-default: all trusted certificates that came with the device.
 - user-stored: all trusted certificates that were installed by the user.
 - all-stored: all trusted certificates stored on the device.
- The local identity will be the common name or the subject alternate name of the client own certificate.
- The remote identity will be the common name or the subject alternate name of the server own certificate.



The configuration of the following parameter is optional and can be omitted.

Match

- Indicates which packets will be allowed to go through the tunnel. You may specify local and remote traffic selectors (subnets).
- Local traffic selector indicates a subnet located on this side of the tunnel. Only packets from/to an IP address within this subnet may be sent/received on the tunnel.

- Remote traffic selector indicates a subnet located on the other side of the tunnel. Only packets from/to an IP address within this subnet will be sent/receive on the tunnel.
- Default for local and remote traffic selector is the any address. You might need to specify more specific traffic selectors depending of the server configuration.
- IKEv1 support one local and one remote traffic selector.
- IKEv2 has no such limitation.

Creating an IPsec tunnel and entering configuration mode

This procedure describes how to create an IPsec tunnel and enter the IPsec tunnel configuration mode.

Mode: Administrator exec

Step	Command	Purpose
1	<code>node(cfg)# tunnel ipsec <name></code>	Enter the IPsec tunnel configuration mode. Mandatory.
	<code>node(tn-ipsec)[name]#</code>	

Configuring the IPsec tunnel

This procedure describes how to create an IPsec tunnel and enter the IPsec tunnel configuration mode.

Mode: tunnel ipsec

Step	Command	Purpose
2	<code>node(tn-ipsec)[name]# local <ctx name> <if name> <label></code>	Set the local IP address. Mandatory.
	<code>node(tn-ipsec)[name]# remote <IP address></code>	Set the remote IP address. Mandatory.
	<code>node(tn-ipsec)[name]# bind <ctx name> <if name></code>	Bind the resulting tunnel to a specific interface. Mandatory.
	<code>node(tn-ipsec)[name]# use profile ipsec <profile></code>	Indicate which IPsec profile must be used. Mandatory.

Entering the authentication configuration mode

This procedure describes how to enter the authentication configuration mode.

Mode: tunnel ipsec

Step	Command	Purpose
3	<code>node(tn-ipsec)[name]# authentication</code>	Enter the authentication configuration mode. Mandatory.
	<code>node(tn-ipsec-authentication)[name]#</code>	

Configuring the authentication

This procedure describes how to configure the authentication.

Mode: authentication

Step	Command	Purpose
4	<code>node(tn-ipsec-auth)[name]# local-id <identity></code>	Set the local identity. Mandatory.
	<code>node(tn-ipsec-auth)[name]# remote-id <identity></code>	Set the remote identity. Mandatory.
	<code>node(tn-ipsec-auth)[name]# method <method></code>	Set the authentication method, pre-shared-key or public-key. Optional. Default is pre-shared key.

If the authentication method is pre-shared key, the pre-shared key must also be configured.

Mode: authentication

Step	Command	Purpose
5a	<code>node(tn-ipsec-auth)# pre-shared-key <key></code>	Set the pre-shared key. Mandatory if method is pre-shared-key.

If the authentication method is public-key, the PKI parameters must also be configured.

Mode: authentication

Step	Command	Purpose
5b	<code>node(tn-ipsec-auth)# own-certificate <own-certificate></code>	Add one own-certificate to the certificate chain. Mandatory if method is public-key.
	<code>node(tn-ipsec-auth)[name]# private-key <private-key></code>	Set the private key associated with the own-certificate. Mandatory if method is public-key.
	<code>node(tn-ipsec-auth)# trusted-certificate <trusted-certificate></code>	Add one trusted certificate to the list of trusted certificates. Mandatory if method is public-key.

Configure traffic selectors

A traffic selector is a filter which controls which packets can transit through an IPsec tunnel. There are local and remote traffic selectors. A packet is allowed through the tunnel if its local and remote IP addresses match one of the local and remote traffic selectors. At the conceptual level, you may visualize a traffic selector as a subnet. By default, the any address is configured for both the local and remote traffic selectors. Some IPsec servers are very restrictive during the traffic selectors negotiation and will not allow the establishment of a tunnel if the offered traffic selectors are wider than what the servers have been configured with. When this is the case, you will need to configure valid local and remote traffic selectors.

IKEv1 supports only one local traffic selector and one remote traffic selector. IKEv2 has no such limitation.

Entering configuration mode

This procedure describes how to enter the IPsec tunnel configuration mode.

Mode: Administrator exec

Step	Command	Purpose
1	<code>node(cfg)# tunnel ipsec <name></code>	Enter the IPsec tunnel configuration mode. Optional.
	<code>node(tn-ipsec)[name]#</code>	

Configuring the traffic selectors

This procedure describes how to configure the local and remote traffic selectors.

Mode: tunnel ipsec

Step	Command	Purpose
2	<code>node(tn-ipsec)[name]# match local <a.b.c.d/m></code>	Add a new local traffic selector. Optional. Default is 0.0.0.0/0.
	<code>node(tn-ipsec)[name]# match remote <a.b.c.d/m></code>	Add a new remote traffic selector. Optional. Default is 0.0.0.0/0.

Configure road warrior vs site-to-site VPN

Trinity supports two different deployment use cases:

- Site-to-site
- Road warrior

Site-to-site deployments are used to establish a secure communication channel between subnets located in different geographical location and that needs to be tied together via the internet. It does not require any other configuration that what has been described in the previous chapter.

Road warriors are remote devices who need secure access to an organization infrastructure. An example would be an employee who is at home and needs to access a document on its employer server. When deploying a road warrior, a new IP address of type "ipsec" must be added to the IP interface bound to by the IPsec tunnel. The client requests an IP address but in the end, it is the server who decides and allocates the final IP address. There is no guaranty that the requested IP address will be granted.

The presence or the absence of an IP address of type "ipsec" is the only determining factor that indicates if we are in the presence of a Site-to-site or a Road warriors deployment.

Figure 40 on page 325 shows a site-to-site IPsec tunnel being established between a branch office and the main office. Additionally how can also see a road warrior (mobile user) connecting to the main office using an IPsec tunnel. Users Alice, Bob and Road warrior have access to the main office servers as if they were on-site.

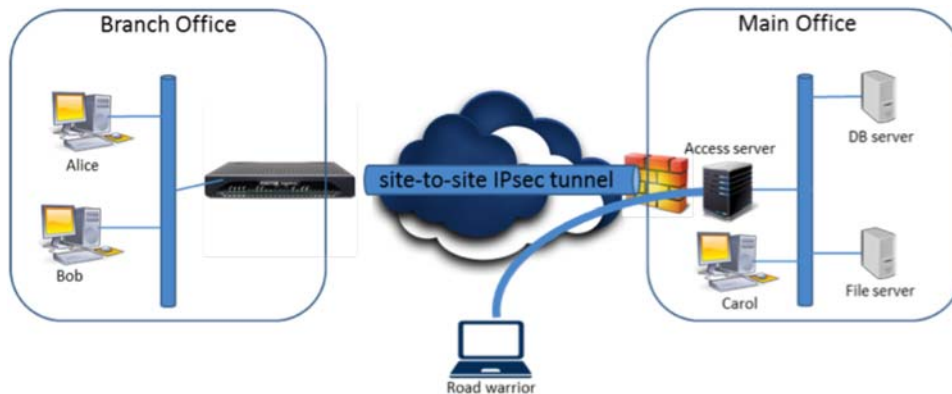


Figure 40. Site-to-site IPsec tunnel being established between a branch office and the main office

Mode: Administrator exec

Step	Command	Purpose
1	<code>node(cfg)# context ip ROUTER</code>	Enter the context IP ROUTE configuration mode.
	<code>node(ctx-ip)[ROUTER]# interface <ipsec-interface></code>	Enter the IP interface configuration mode.
	<code>node(if-ip)[ipsec-interface]# ipaddress <name> ipsec request <a.b.c.d></code>	Add a new IP address of type ipsec on the interface on which the IPsec tunnel is bound.

Examples

This series of examples taken together allow the establishment of an IP tunnel. Section “[IP interface configuration](#)” and section “[IPsec tunnel configuration](#)” each propose two options. Only one of the options must be selected.

IPsec profile configuration

```

profile ipsec profile-ipsec-example
  phase-1-ike
    diffie-hellman-group modp-3072
    encryption-algorithm aes-256
    integrity-algorithm sha1-160-96
    key-lifetime 3600 seconds
  phase-2-ipsec
    diffie-hellman-group modp-3072
    encryption-algorithm aes-256
    integrity-algorithm sha1-160-96
    key-lifetime 600 seconds
    key-lifetime 1024 kilobytes
    key-lifetime 1000 packets

```

IP interface configuration

Site-to-site:

```
context ip
  interface interface-ipsec-example
```

OR

Road warrior:

```
context ip
  interface interface-ipsec-example
    ipaddress ipaddress-ipsec-example ipsec request 10.10.10.10
```

IPsec tunnel configuration

Pre-shared key authentication:

```
tunnel ipsec tunnel-ipsec-example
  local ROUTE WAN WAN
  remote 192.168.1.145
  bind interface ROUTE interface-ipsec-example
  use profile ipsec profile-ipsec-example
  match local 10.3.0.0/24
  match remote 10.4.0.0/24
  authentication
    local-id client
    remote-id server
    pre-shared-key "this is the pre-shared key"
  no shutdown
```

OR

Public key authentication:

```
tunnel ipsec tunnel-ipsec-example
  local ROUTE WAN WAN
  remote 192.168.1.145
  bind interface ROUTE interface-ipsec-example
  use profile ipsec profile-ipsec-example
  match local 10.3.0.0/24
  match remote 10.4.0.0/24
  authentication
    method public-key
    local-id client.com
    remote-id server.com
    own-certificate pki:certificate/client.cert.pem
    own-certificate pki:certificate/client-ca.cert.pem
    own-certificate pki:certificate/root.cert.pem
    private-key pki:private-key/client.key.pem
    trusted-certificate pki:trusted-certificate/root.cert.pem
  no shutdown
```

Troubleshooting

The following commands are useful to examine the configuration and the status of IPsec.

Mode: operator

Step	Command	Purpose
	node # show profile ipsec <name>	Show the configuration for an IPsec profile.
	node # show tunnel ipsec <name> config	Show the configuration for an IPsec tunnel.
	node # show tunnel ipsec <name> status	Show the status information for an IPsec tunnel.

It is possible to trace run-time information on IPsec.

Mode: operator

Step	Command	Purpose
	node # [no] trace ipsec	Trace configuration changes and supervision of IPsec daemon.

Diffie-Hellman groups and encryption/integrity algorithms

Here is the list of all Diffie-Hellman groups and encryption/integrity algorithm that are supported:

Diffie-Hellman groups

- modp-768, modp-1024, modp-1536, modp-2048, modp-3072, modp-4096, modp-6144 and modp-8192.
- Anything under 2048 is considered insecure.

Encryption algorithms

- 3des, aes-128, aes-192 and aes-256.
- 3des is now considered insecure.

Integrity algorithms

- md5-128, md5-128-96, sha1-160, sha1-160-96, sha2-256, sha2-384 and sha2-512.
- md5 and sha1 algorithms are considered insecure.
- There is some limitation concerning the usage of some of the integrity algorithms
 - md5-128 and sha1-160 are not supported in phase 1.
 - md5-128 and sha1-160 are not supported in phase 2 if the key exchange protocol is "ikev1" or "all"

Chapter 25 **OpenVPN Configuration**

Chapter contents

Introduction	330
OpenVPN Configuration Task List (Quick Start)	331
Configuring NTP	331
Configuring OpenVPN Tunnel	331
OpenVPN Configuration Task List (Full Reference)	332
Configuring NTP ¶	332
Creating a Public-Key Infrastructure (PKI) ¶	332
Creating a Certificate Authority (CA) (Server Only)	333
Obtaining the Keys and Certificates (Server)	333
Obtaining the Keys and Certificates (Client)	333
Configuring the TLS Profile	334
Configuring OpenVPN Tunnel	335
Debugging OpenVPN	336
Sample Configurations	338
Common Configuration	338
Application 1: Point-to-point	339
Application 2: Point-to-multipoint	339

Introduction

This chapter describes how to configure an OpenVPN tunnel in Trinity 3. This may be used to implement a virtual private network between two Trinity 3 devices, or between a Trinity 3 device and PC running Windows, MAC OS, or Linux.

A virtual private network (VPN) is a private data network that uses the public telecommunications infrastructure, maintaining privacy through the use of a tunneling protocol and security procedures.

OpenVPN may be used in one of two modes, point-to-point (see [figure 41](#)) or point-to-multipoint (see [figure 42](#)).

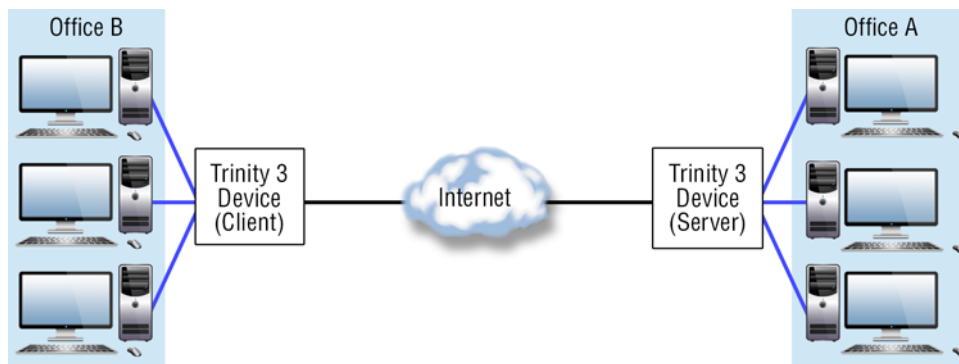


Figure 41. Point-to-point

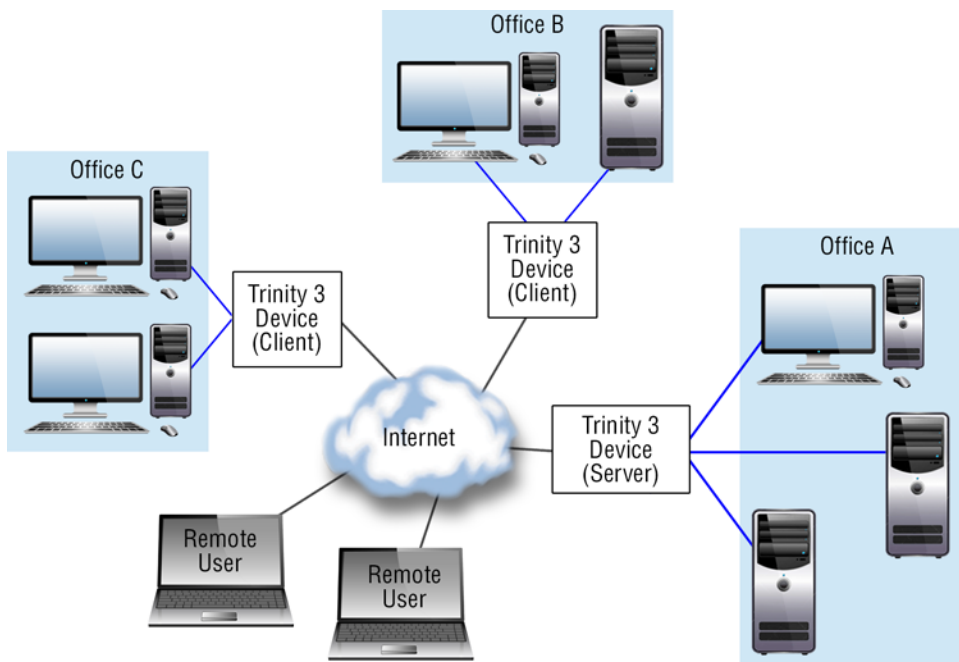


Figure 42. Point-to-Multipoint

The difference is that in point-to-point mode, the server only allows a single client to connect, and specifies the IP address from which that client may connect; whereas in point-to-multipoint mode the server allows multiple clients to connect.

Note In point-to-multipoint mode, the clients may communicate with each other, but all their communication is through the server.

The following section describes the tasks involved in configuring an OpenVPN tunnel.

OpenVPN Configuration Task List (Quick Start)

This section describes the minimal steps required to configure a basic point-to-multipoint OpenVPN setup. You may then refer to section “[OpenVPN Configuration Task List \(Full Reference\)](#)” on page 332 to tweak the configuration.

Note These steps involve the OpenVPN server generating each of the clients' private keys, which are then imported into the client. As its name implies, the private key must remain private, or the tunnel's security will be compromised, so be sure to only transmit the key using secure means. For even more security, you may follow the instructions in “[OpenVPN Configuration Task List \(Full Reference\)](#)” on page 332 instead, in which case the clients generate their own private keys, and so the private keys are never transmitted over the network.

Configuring NTP

Part of certificate verification is checking the certificate's validity dates which requires the device to have reasonably accurate time. See chapter 33, “[SNTP Client Configuration](#)” on page 390 for details.

Configuring OpenVPN Tunnel

This section describes the commands used to create and configure an OpenVPN tunnel.

To configure the server, do the following steps:

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg)#tunnel openvpn <name></code>	Create OpenVPN tunnel <i>name</i> .
2	<code>node(openvpn)[<name>]#auto-init as-server</code>	Create a PKI for the server, configure a TLS profile, bind the tunnel to the TLS profile, and configure the tunnel in server mode.
3	<code>node(openvpn)[<name>]#local <context-ip> <interface> <label></code>	Listen for connections on a particular IP address.
4	<code>node(openvpn)[<name>]#no shutdown</code>	Enable the OpenVPN tunnel.

For each client, follow these steps on the server:

Mode: tunnel openvpn <name>

Step	Command	Purpose
1	node(openvpn)[<name>]#auto-create client <client> {trinity-format raw-format}	<p>Extend the PKI for the client and output commands to be pasted into the client's CLI.</p> <p>You may run this command multiple times for the same client. It will only extend the PKI the first time. After that, it will only output the commands to be pasted into the client's CLI.</p> <p>If you specify <i>raw-format</i>, the output can be pasted into a *.ovpn file on your PC to allow your PC to connect.</p>

For each client, follow these steps on the client:

Mode: tunnel openvpn <name>

Step	Command	Purpose
1	node(openvpn)[<name>]#no shutdown	Enable the OpenVPN tunnel.

OpenVPN Configuration Task List (Full Reference)

This section lists the full set of commands available for configuring OpenVPN.

You may either follow this task list on its own as a secure and flexible way to configure your OpenVPN setup, or you may optionally refer to it after you have completed the Quick Start in order to tweak your setup.

Configuring NTP¶

Part of certificate verification is checking the certificate's validity dates which requires the device to have reasonably accurate time. See chapter 33, “SNTP Client Configuration” on page 390 for details.

Creating a Public-Key Infrastructure (PKI)¶

OpenVPN relies on a PKI to authenticate OpenVPN tunnel endpoints. You may follow the steps in this section for an example, or you may see chapter 35, “Public-Key Infrastructure (PKI)” on page 408 for more information.

Creating a Certificate Authority (CA) (Server Only)

This section contains an example of creating a certificate authority. These steps only apply to the OpenVPN server. The clients do not need to do this.

Step	Command
1	generate pki:certificate-authority/VPN country US state Maryland locality Gaithersburg organization "Patton Electronics Company" organization-unit "Software Engineering" common-name "VPN Certificate Authority"

Obtaining the Keys and Certificates (Server)

This section contains an example of obtaining the private key, signed certificate, and CA certificate. These steps differ for the OpenVPN server and client because the CA resides on the server so it does not need to copy files. These commands are to be executed on the OpenVPN server.

Generate our private key and certificate request:

Step	Command	Purpose
1	generate pki:private-key/VPN key-length 2048	A lesser key length may be used for decreased security and increased performance
2	generate pki:certificate-request/VPN private-key pki:private-key:VPN country US state Maryland locality Gaithersburg organization "Patton Electronics Company" organizational-unit "Software Engineering" common-name <serial-number>	

Sign our certificate:

Step	Command
1	generate pki:certificate/VPN certificate-request pki:certificate-request/VPN certificate-authority pki:certificate-authority/VPN certificate-type server

Obtaining the Keys and Certificates (Client)

This section contains an example of obtaining the private key, signed certificate, and CA certificate. These steps differ for the OpenVPN server and client because the CA resides on the server so it does not need to copy files. These commands are to be executed on each OpenVPN client.

Generate our private key and certificate request:

Step	Command	Purpose
1	generate pki:private-key/VPN key-length 2048	A lesser key length may be used for decreased security and increased performance

Step	Command	Purpose
2	generate pki:certificate-request/VPN private-key pki:private-key:VPN country US state Maryland locality Gaithersburg organization "Patton Electronics Company" organizational-unit "Software Engineering" common-name <serial-number>	
3	export pki:certificate-request/VPN	Copy the output and email it to the server's administrator.

Sign the client's certificate (on the OpenVPN server):

Step	Command	Purpose
1	import pki:certificate-request/VPN_CLIENT<x>	Copy and paste the output of the export command above.
2	generate pki:certificate/VPN_CLIENT<x> certificate-request pki:certificate-request/VPN_CLIENT<x> certificate-authority pki:certificate-authority/VPN certificate-type client	
3	export pki:certificate/VPN_CLIENT<x>	Copy the output and email it to the client's administrator.
4	export pki:trusted-certificate/VPN	Copy the output and email it to the client's administrator.

Import the CA's public certificate and our signed certificate (on the OpenVPN client):

Step	Command	Purpose
1	import pki:certificate/VPN	Copy and paste the output of the export command above.
2	import pki:trusted-certificate/VPN	Copy and paste the output of the export command above.

Configuring the TLS Profile

This section contains an example of configuring the TLS profile that the OpenVPN tunnel will use. These commands are to be executed on each OpenVPN tunnel endpoint.

Create and configure the TLS profile:

Step	Command	Purpose
1	profile tls VPN	
2	private-key pki:private-key/VPN	
3	own-certificate 1 pki:certificate/VPN	
4	trusted-certificate pki:trusted-certificate/VPN	

Step	Command	Purpose
5	certificate-revocation-list pki:certificate-revocation-list/VPN	Optional - for server only; this allows the server to reject connections from clients whose certificates have been revoked
6	diffie-hellman-parameters pki:diffie-hellman-parameters/DEFAULT-4096	Only required for server; also a shorter prime of 1024 or 2048 may be used for decreased security and increased performance
7	require certificate-type server	Optional - for client only; this will reject the peer's pki:certificate unless it was signed with certificate-type server

Leave all other TLS profile parameters as their default value.

Configuring OpenVPN Tunnel

This section describes the commands used to create and configure an OpenVPN tunnel.

Mode: Configure

Step	Command	Purpose
1	node(cfg)#tunnel openvpn <name>	Create OpenVPN tunnel <i>name</i> .
2	node(openvpn)[VPN]#mode {server client}	Configure the mode. One endpoint of the tunnel must be a server. The rest must be clients.
3	node(openvpn)[VPN]#type bridged	Configure the encapsulation type. Currently, Trinity 3 only supports bridged type, that is, encapsulation of Ethernet frames.
4	node(openvpn)[VPN]#local [<context-ip> <interface> <label>] [port <0..65535>]	(Optional) Listen for connections on a particular IP address and/or a non-standard port. By default, OpenVPN listens on all IP addresses on port 1194.
5	node(openvpn)[VPN]#remote <A.B.C.D> [port <0..65535>]	(Optional in server mode) Connect to an OpenVPN tunnel endpoint that is listening on the specified IP address and port. If no port is specified, the default port of 1194 will be used. If in server mode and not specified, the tunnel will be point-to-multipoint, allowing multiple clients to connect.
6	node(openvpn)[VPN]#use profile tls <name>	Use the keys/certificates specified by TLS profile <i>name</i> .

Step	Command	Purpose
7	node(openvpn)[VPN]#no shutdown	Enable the OpenVPN tunnel.

Debugging OpenVPN

Common problems that result in a failure to connect are:

- **Incorrect time:** If the time is wrong, the certificate verification may fail when the certificate's validity date is checked.
- **The local IP address is not up:** If the OpenVPN tunnel is bound to a local IP address, then it cannot start until that address comes up.

The following commands may be used to debug OpenVPN:

Mode: Operator exec

Step	Command	Purpose
1	node>show tunnel openvpn <name>	Show the tunnel configuration and status
2	node>trace openvpn debug detail 0	Show the logs from the OpenVPN process. This can provide details on why a connection fails to establish.
3	node>show system time	Ensure that the time is correct so that the device can correctly verify certificates.

This is sample output of *show tunnel openvpn <name>*:

```
vpn openvpn VPN
=====
Mode           : server
Type           : bridged
Local          : ROUTER.WAN.WAN:1194 (up, 10.0.1.2)
Remote        :
TLS Profile    : VPN
Administrative State : Up
Bound Bridge   : VPN
Connections    : Client2 (10.0.3.2:1194)
                Client1 (10.0.2.2:1194)
Operational State : Up
Hardware Address : ae:76:e0:b7:f2:7f
MTU            : 1500
ARP            : enabled
Multicast      : enabled
Rx Statistics   : 780 bytes in 10 packets
                0 errors 0 drops, 0 overruns
                0 multicast packets
Tx Statistics   : 685 bytes in 8 packets
                0 errors 0 drops, 0 collisions 0 carrier errors
```

Note Verify that the Local IP address is up, and that there are two connections.

This is sample output from *trace openvpn debug detail 0* while the OpenVPN process initializes.

```
16:54:40.000 OPENVPN # [VPN] Process: OpenVPN 2.3.8 arm-buildroot-linux-gnueabi
[SSL (OpenSSL)] [LZO] [EPOLL] [MH] [IPv6] built on Oct 30 2015
16:54:40.000 OPENVPN # [VPN] Process: library versions: OpenSSL 1.0.11 15 Jan
2015, LZO 2.06
16:54:40.000 OPENVPN # [VPN] Process: NOTE: the current --script-security setting
may allow this configuration to call user-defined scripts
16:54:40.000 OPENVPN # [VPN] Process: Diffie-Hellman initialized with 2048 bit
key
16:54:40.000 OPENVPN # [VPN] Process: WARNING: file '/flash/pki/private-key/
DEFAULT' is group or others accessible
16:54:40.000 OPENVPN # [VPN] Process: Socket Buffers: R=[163840->131072]
S=[163840->131072]
16:54:40.000 OPENVPN # [VPN] Process: TUN/TAP device openvpn0 opened
16:54:40.000 OPENVPN # [VPN] Process: TUN/TAP TX queue length set to 100
16:54:40.000 OPENVPN # [VPN] Process: UDPv4 link local (bound):
[AF_INET]10.0.1.2:1194
16:54:40.000 OPENVPN # [VPN] Process: UDPv4 link remote: [undef]
16:54:40.000 OPENVPN # [VPN] Process: MULTI: multi_init called, r=256 v=256
16:54:40.000 OPENVPN # [VPN] Process: Initialization Sequence Completed
```

This is sample output from *trace openvpn debug detail 0* while a client connects.

```
16:54:40.000 OPENVPN # [VPN] Process: 10.0.2.2:1194 TLS: Initial packet from
[AF_INET]10.0.2.2:1194, sid=645bd9d0 7dae962e
16:54:41.000 OPENVPN # [VPN] Process: 10.0.2.2:1194 VERIFY OK: depth=1, C=US,
ST=Maryland, L=Gaithersburg, O=Patton Electronics Company, OU=Software Engineering,
CN=CA, emailAddress=stochs@patton.com
16:54:41.000 OPENVPN # [VPN] Process: 10.0.2.2:1194 VERIFY OK: depth=0, C=US,
ST=Maryland, O=Patton Electronics Company, OU=Software Engineering, CN=Client1,
emailAddress=stochs@patton.com
16:54:41.000 OPENVPN # [VPN] Process: 10.0.2.2:1194 Data Channel Encrypt: Cipher
'BF-CBC' initialized with 128 bit key
16:54:41.000 OPENVPN # [VPN] Process: 10.0.2.2:1194 Data Channel Encrypt: Using
160 bit message hash 'SHA1' for HMAC authentication
16:54:41.000 OPENVPN # [VPN] Process: 10.0.2.2:1194 Data Channel Decrypt: Cipher
'BF-CBC' initialized with 128 bit key
16:54:41.000 OPENVPN # [VPN] Process: 10.0.2.2:1194 Data Channel Decrypt: Using
160 bit message hash 'SHA1' for HMAC authentication
16:54:41.000 OPENVPN # [VPN] Process: 10.0.2.2:1194 Control Channel: TLSv1,
cipher TLSv1/SSLv3 DHE-RSA-AES256-SHA, 2048 bit RSA
16:54:41.000 OPENVPN # [VPN] Process: 10.0.2.2:1194 [Client1] Peer Connection
Initiated with [AF_INET]10.0.2.2:1194
16:54:41.000 OPENVPN # [VPN] Process: Client1/10.0.2.2:1194 OPTIONS IMPORT: read-
ing client specific options from: /tmp/openvpn_c-
c_0c9fe0f2eda9c4b4c46cdac8e184e4a4.tmp
16:54:41.000 OPENVPN # [VPN] Process: Client1/10.0.2.2:1194 MULTI: no dynamic or
static remote --ifconfig address is available for Client1/10.0.2.2:1194
16:54:41.000 OPENVPN # [VPN] Process: Connected to Client1 at 10.0.2.2:1194
16:54:41.000 EVENT # Link vpn openvpn VPN up
```

Sample Configurations

Common Configuration

All devices in all the applications below have the following common configuration:

```
# Enable NTP, because we need the correct time to verify certificates.
ntp
  server 0.pool.ntp.org
  server 1.pool.ntp.org
  server 2.pool.ntp.org
  server 3.pool.ntp.org
no shutdown

# Specify the keys/certificates to use. The private-key and own-certificate,
# despite having the same name, differ in content for each device. The trusted-
# certificate and diffie-hellman-parameters are the same. See section "Creating
# a Public-Key Infrastructure (PKI)" for details.
profile tls VPN
  private-key pki:private-key/VPN
  own-certificate 1 pki:certificate/VPN
  trusted-certificate pki:trusted-certificate/VPN
  diffie-hellman-parameters pki:diffie-hellman-parameters/DEFAULT-4096

# Each device must have a connection to the Internet.
context ip ROUTER

  interface WAN
    ipaddress WAN <wan-ip>/<mask>

  routing-table DEFAULT
    route default gateway <gateway-ip>

context bridge

  bridge-group VPN
    no shutdown

tunnel openvpn VPN
  local ROUTER WAN WAN
  use profile tls VPN
  bind bridge-group VPN
  no shutdown

# Ethernet 0/0 is connected to the Internet.
port ethernet 0 0
  bind interface ROUTER WAN
  no shutdown

# Ethernet 0/1 is connected to the LAN. All traffic from the VPN is bridged
# onto the LAN, and vice versa.
port ethernet 0 1
  bind bridge-group VPN
  no shutdown
```

Application 1: Point-to-point

This configures the application shown in [figure 41](#) on page 330.

In addition to the common configuration above, add the following configuration for the Server:

```
# The <wan-ip> from the common configuration above must be a public IP address,  
# so that the client can connect to it.  
  
# Specify the remote to make this a point-to-point connection.  
tunnel openvpn VPN  
  mode server  
  remote <client-wan-ip>
```

and for the Client:

```
# The <wan-ip> from the common configuration above must be a public IP address,  
# so that the server can connect to it.  
  
tunnel openvpn VPN  
  remote <server-wan-ip>
```

Application 2: Point-to-multipoint

This configures the application shown in [figure 42](#) on page 330.

In addition to the common configuration above, add the following configuration for the Server:

```
# The <wan-ip> from the common configuration above must be a public IP address,  
# so that the clients can connect to it.  
  
# Leave the remote unspecified to make this a point-to-multipoint connection.  
tunnel openvpn VPN  
  mode server  
and for Client1 and Client2:  
# The <wan-ip> from the common configuration above need not be a public IP  
# address, so we will initiate the connection to the server, and not vice versa.  
  
tunnel openvpn VPN  
  remote <server-wan-ip>
```

Chapter 26 **Generic Routing Encapsulation (GRE) Configuration**

Chapter contents

Introduction	341
GRE Configuration Task List	341
Configuring GRE Tunnel	341
Debugging GRE Tunnel	342
Sample Configurations	343
Common Configuration	343

Introduction

This chapter describes how to configure generic routing encapsulation (a GRE tunnel) in Trinity 3. GRE is a method of encapsulating one protocol within another protocol. This implementation only supports encapsulating IPv4 in IPv4, which may be used to implement a virtual private network between two Trinity 3 devices, or between a Trinity 3 device and a Cisco device. However, it is important to note that the data is not encrypted, so it should not be sent across the Internet.

The GRE tunnel is set up between two devices, as shown in [figure 43](#):

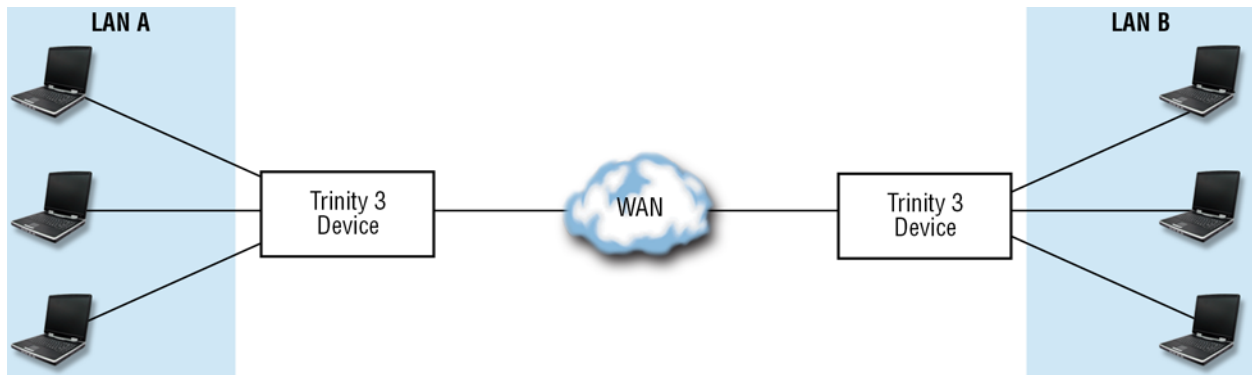


Figure 43. GRE Tunnel across a WAN

Both devices must be set up to enable the GRE tunnel. When configured correctly, traffic sent from a device on LAN A to a device on LAN B will reach its destination (and vice-versa), and is encapsulated with GRE protocol while traversing the WAN. Trinity 3 devices encapsulate as data enters the WAN, and unencapsulate as data exits the WAN.

Note GRE is a peer to peer protocol; there is no client or server in this configuration.

The following sections describe the tasks involved in configuring a GRE tunnel.

GRE Configuration Task List

Configuring GRE Tunnel

This section describes the commands used to create and configure a GRE tunnel. Note that additional configuration of a routing-table and ip interfaces is required outside of the GRE commands, as shown in the sample configuration.

Mode: configure

Step	Command	Purpose
1	<code>node(cfg)#tunnel gre <name></code>	Create GRE tunnel <i>name</i> .

Step	Command	Purpose
2	<code>node(gre)[GRE]#remote <A.B.C.D></code>	(Required) Connect to a remote tunnel endpoint that is listening on the specified IP address. This is the WAN ip address of the opposing Trinity 3 device.
3	<code>node(gre)[GRE]#bind interface [<context-ip> <interface>]</code>	(Required) Bind the tunnel to the specified IP Interface. Only the other end of the tunnel can access this interface, it is not bound to a port ethernet, and is used to route the appropriate traffic through the tunnel. This is usually an IP Interface called <i>VPN</i> .
4	<code>node(gre)[GRE]#local [<context-ip> <interface> <label>]</code>	(Optional) Listen for connections on a particular IP address. This is usually the WAN ip address label of the local Trinity 3 device. By default, GRE listens on all IP addresses.
5	<code>node(gre)[GRE]#no shutdown</code>	Enable the GRE tunnel.

Debugging GRE Tunnel

This section describes the commands used to create and configure a GRE tunnel. Note that additional configuration of a routing-table and ip interfaces is required outside of the GRE commands, as shown in the sample configuration

A common problem that results in a failure to connect is:

- **The local IP address is not up:** If the GRE tunnel is bound to a local IP address, then it cannot start until that address comes up.

The following command may be used to debug GRE:

Mode: Operator execution

Step	Command	Purpose
1	<code>node>show tunnel gre <name></code>	Show the tunnel configuration and status.

This is sample output of show tunnel gre <name>:

```
gre GRE
=====

Local           : ROUTER.WAN.WAN:1194 (up, 1.1.1.1)
Remote          : 1.1.1.2
Bind            : VPN
Administrative State : Up
Operational State  : Up
Hardware Address  : ae:76:e0:b7:f2:7f
```

```

MTU                : 1500
ARP                : enabled
Multicast          : enabled
Rx Statistics      : 780 bytes in 10 packets
                   : 0 errors 0 drops, 0 overruns
                   : 0 multicast packets
Tx Statistics      : 685 bytes in 8 packets
                   : 0 errors 0 drops, 0 collisions 0 carrier errors

```

Note The Local IP address is up.

Sample Configurations

Common Configuration

These configurations create a GRE Tunnel between two Trinity 3 devices across a WAN, where Device A is on LAN A, and Device B is on LAN B. See [figure 44](#) on page 344, the diagram of this sample configuration.

Trinity 3 Device A, on LAN A:

```

context ip ROUTER

interface LAN
  ipaddress LAN 172.16.1.1/24

interface WAN
  ipaddress WAN 1.1.1.1/30

# Note that the VPN interface and routing-table compliment Device B's:
# Device B's routing-table gateway is this VPN interface.
# Device B's VPN interface is this routing-table's gateway.

interface VPN
  ipaddress VPN 10.0.0.1/30
x
routing-table DEFAULT
  route 172.16.2.0/24 gateway 10.0.0.2

port ethernet 0 0
  bind interface ROUTER LAN
  no shutdown

port ethernet 0 1
  bind interface ROUTER WAN
  no shutdown

tunnel gre GRE
  local ROUTER WAN WAN # optional
  remote 1.1.1.2
  bind interface ROUTER VPN
  no shutdown

```

Trinity 3 Device B, on LAN B.

```

context ip ROUTER

interface LAN
  ipaddress LAN 172.16.2.1/24

interface WAN
  ipaddress WAN 1.1.1.2/30

interface VPN
  ipaddress VPN 10.0.0.2/30

routing-table DEFAULT
  route 172.16.1.0/24 gateway 10.0.0.1

port ethernet 0 0
  bind interface ROUTER LAN
  no shutdown

port ethernet 0 1
  bind interface ROUTER WAN
  no shutdown

tunnel gre GRE
  local ROUTER WAN WAN # optional
  remote 1.1.1.1
  bind interface ROUTER VPN
  no shutdown

```

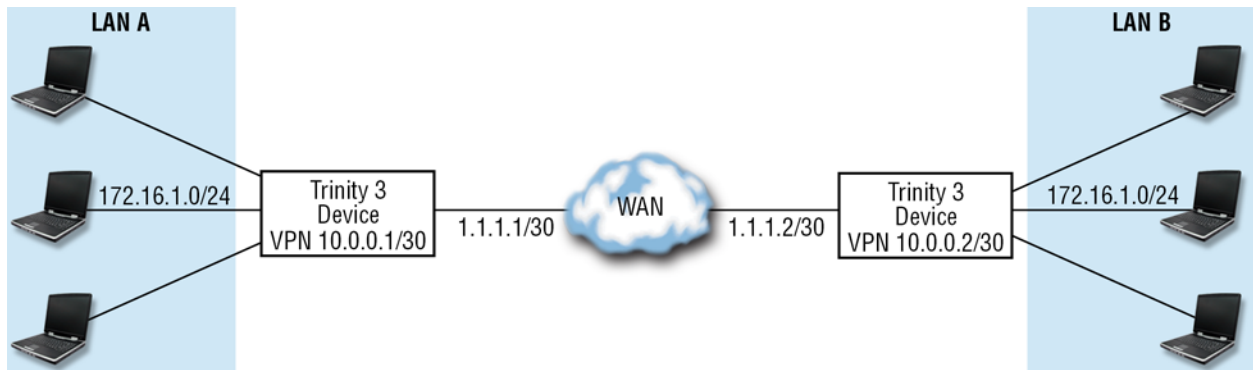


Figure 44. Diagram of sample configuration.

Chapter 27 **Border Gateway Protocol (BGP) Configuration**

Chapter contents

Introduction	346
About Border Gateway Protocol (BGP)	346
BGP configuration task list	346
Starting BGP	346
Configure BGP general behavior	347
Always compare MED	347
Deterministic MED	347
Log neighbor changes	348
Graceful restart	348
Timers	348
Distance	349
Redistribute	349
Configure BGP Networks	349
Configure BGP Neighbors	350
Configure BGP neighbor behavior	350
Configure BGP neighbor route-mappings	351
Configure route-map profile	351
Match rules	352
Set rules	353
Examples	354
Troubleshooting	355

Introduction

This chapter gives an overview of the Border Gateway Protocol (BGP), and describes how to use BGP commands to configure Trinity. Trinity supports Border Gateway Protocol version 4 (BGP-4).

About Border Gateway Protocol (BGP)

BGP is an autonomous system (AS) routing protocol that is used in a network or group of networks to share routing information between Internet Service Providers (ISPs). BGP speakers exchange routing information when a TCP connection is first established between them. BGP routing updates only show the best routing path to the destination network.

BGP attributes are properties of associated routes that determine the best route to a destination when multiple paths exist. The BGP speaker will determine the best route based on the weight and local preference attributes.

BGP configuration task list

To configure BGP, perform the tasks in the following sections:

- Starting BGP
- Configuring BGP general behavior (optional)
- Configuring BGP networks
- Configuring BGP neighbors
- Configuring BGP neighbor behavior (optional)
- Configuring BGP neighbor route-mappings (optional)
- Configuring profile route-map (optional)
- Configuring profile route-map match rules (optional)
- Configuring profile route-map set rules (optional)

Starting BGP

To configure BGP, the Patton device needs to know the autonomous system (AS) where it is part of and the router ID to identify itself within the AS. The router ID must be the highest public IP-address of the Device.

Mode: context ip

Step	Command	Purpose
1	<code>node(ctx-ip)[name]# bgp</code>	Enter the BGP configuration mode.
2	<code>node(ctx-ip)[name]# reset bgp</code>	Reset the complete BGP configuration. When you like to keep the configuration and want only to disable the BGP router then use rather the shutdown command within the BGP mode.

Mode: context ip/bgp

Step	Command	Purpose
1	node(bgp)[name]# as-identifier <number>	Set the mandatory identifier of the AS where this BGP speaker is part of.
2	node(bgp)[name]# router-id <ip-address>	Set the mandatory identifier of the router. This must be the highest public ip-address of the device.
3	node(bgp)[name]# port <number>	Specifies the TCP port on which this BGP router is reachable. Default: 179.
4	node(bgp)[name]# [no] shutdown	Enables or disables the BGP router. Default: disabled.

Configure BGP general behavior

This section contains all commands for specifying the general BGP behavior. These apply for all BGP neighbors and the internal algorithms for selecting routes.

Always compare MED

The MED is one of the parameters which can be used to select the best path among different paths available. This command specifies if the MED is considered for path selection among routes originating from different AS.

Mode: context ip/bgp

Step	Command	Purpose
1	node(bgp)[name]# [no] always-compare-med	Compare MED among routes originating from different AS. Default: disabled.

Deterministic MED

The MED is one of the parameters which can be used to select the best path among different paths available. This command specifies if the MED is considered for path selection among routes originating from different peers from within the same AS.

Mode: context ip/bgp

Step	Command	Purpose
1	node(bgp)[name]# [no] deterministic-med	Compare MED among routes originating from different peers from within the same AS. Default: disabled.

Log neighbor changes

This command is used only for debugging. In combination with tracing it allows to see more details when getting route updates from neighboring peers.

Mode: context ip/bgp

Step	Command	Purpose
1	node(bgp)[name]# [no] log-neighbor-changes	Allows to trace route changes communicated from BGP neighbors. Default: disabled.

Graceful restart

RFC 4724 specifies a graceful restart procedure to minimize the negative impact when a BGP speaker restarts. It applies for both sides, the restarting device and the devices having a BGP connection to the restarting device. When a BGP speaker restarts, all routes originating from that speaker are preserved for a certain time. This gives the restarting router time to establish the connections to its peers again and updating the route information before these are considered invalid.

Trinity communicates a restart-time of 120 seconds, which cannot be changed.

Mode: context ip/bgp

Step	Command	Purpose
1	node(bgp)[name]# [no] graceful-restart [stalepath-time <seconds>]	Enables graceful restart procedure according RFC4724. Default: enabled, with a stalepath time of 360 seconds.

Timers

The timers are used to configure the keepalive mechanism for the connections between BGP peers. The hold time is negotiated between the peers and the lower value of them is used. If one BGP speaker does not receive a message from the peer for the duration of the hold time, that connection is considered broken and will be terminated. The BGP speaker sends a keepalive message after the keepalive time, which is not allowed to be bigger than a third of the hold time.

This global timer values can be overwritten for each neighbor specifically. If there is no timer configured in the neighbor, this global timer values are used.

Mode: context ip/bgp

Step	Command	Purpose
1	node(bgp)[name]# timers keepalive <seconds> holdtime <seconds>	Configure the global values for the hold and keepalive time. The keepalive time must be lower or equal a third of the hold time. Default keepalive time: 60 seconds. Default holdtime: 180 seconds.

Distance

Configures the distance for routes within the routing information base, based according the origin of the route:

- External: routes originating from peers belonging to another AS.
- Internal: routes originating from peers within the AS of the BGP speaker.
- Local: routes originating from this BGP speaker itself.

Mode: context ip/bgp

Step	Command	Purpose
1	node(bgp)[name]# distance external <nbr> internal <nbr> local <nbr>	Configures the distance applied to routes according their origin. Defaults: distance external: 20, distance internal: 200, distance local: 200.

Redistribute

Configures which sources should be considered for routes to be announced to peers. Routes which are added to BGP via a redistribute stay alive even if the route disappeared from the source. This is to prevent route flapping through whole BGP networks.

Mode: context ip/bgp

Step	Command	Purpose
1	node(bgp)[name]# redistribute static	Configures to distribute static routes configured in the routing-table. Default disabled.
2	node(bgp)[name]# redistribute connected	Configures to distribute routes of the locally connected networks on the ip interfaces. The interface needs to be up and have link before the route is added to BGP. Default disabled.

Configure BGP Networks

Manually configures routes which are announced to the BGP peers. The next-hop of these routes is always this router itself.

Mode: context ip/bgp

Step	Command	Purpose
1	node(bgp)[name]# [no] network (<a.b.c.d/m> <a.b.c.d> <mask>)	Configure one or multiple networks to announce to peers.

Configure BGP Neighbors

The neighbors are other BGP speakers to whom this BGP speaker establishes connections and exchanges routing information with them. The name of the neighbor needs to match the ip-address of the peer device which is used to establish the BGP connection.

Mode: context ip/bgp

Step	Command	Purpose
1	node(bgp)[name]# [no] neighbor <a.b.c.d>	Configure one or multiple BGP peers to exchange routing information with.

Mode: bgp neighbor

Step	Command	Purpose
1	node(bgp-neigh)[peer]# remote-as <number>	Configure the AS identifier of which the BGP neighbor is part of.
2	node(bgp-neigh)[peer]# [no] shutdown	Enables or disables the connection to the BGP peer. Default: disabled.

Configure BGP neighbor behavior

This section contains all commands for the behavior specific to the BGP neighbor.

Mode: context ip/bgp

Step	Command	Purpose
1	node(bgp-neigh)[peer]# [no] description "<text>"	Set a description text for better readability of the configuration.
2	node(bgp-neigh)[peer]# [no] ebgp-multihop [<hops>]	Specifies that the neighbor from an external AS is more than one hop away from this BGP speaker. The optional hops parameter can be used to limit the maximum number of hops it is away. Default: disabled.
3	node(bgp-neigh)[peer]# port <number>	Specifies the TCP port on which the BGP neighbor is reachable. Default: 179.
4	node(bgp-neigh)[peer]# [no] next-hop-self	Overwrite the next-hop parameter with our own ip-address before forwarding them to the neighbor within an external AS. Default: disabled.
5	node(bgp-neigh)[peer]# [no] default-originate	Enables to send default routes toward that peer. Default: disabled

Step	Command	Purpose
6	node(bgp-neigh)[peer]# [no] maximum-prefix <number>	Specifies the maximum number of routes received from this peer. If this maximum is exceeded, then the connection is closed and all routes are rejected. Default: disabled.
7	node(bgp-neigh)[peer]# [no] timers keepalive <seconds> holdtime <seconds>	Overwrites the global configured values for the hold and keepalive time only for this neighbor. The keepalive time must be lower or equal a third of the hold time. Default: use the globally configured timer values.

Configure BGP neighbor route-mappings

The usage of a route-map profile allows to apply filters on which routes are processed and or to modify parameters of processed routes. As a prerequisite the profile must be created first before it can be used here.

Mode: context ip/bgp

Step	Command	Purpose
1	node(bgp-neigh)[peer]# [no] use profile route-map <profile-name> in	Specifies the profile to filter or modify routes received from this neighbor.
2	node(bgp-neigh)[peer]# [no] use profile route-map <profile-name> out	Specifies the profile to filter or modify routes which are sent to that neighbor.

Configure route-map profile

A route-map profile allows to apply filters on routes and or to modify parameters of processed routes.

Mode: context ip/bgp

Step	Command	Purpose
1	node(cfg)[name]# [no] profile route-map <profile-name>	Creates a new or deletes an existing route-map profile.

A route-map profile consists of a sequence of route-mappings, which are passed in the indexed order in the profile. The first route-mapping whose match rules match to the processed route, is then executed for that route. The action of the route-mapping can be:

- **Permit:** the route is permitted and all the according set statements of the route-mapping are executed on the route. In the outgoing direction it is then sent to the peer or in incoming direction the route is then accepted into the routing information base.

- **Deny:** the route is ignored: in outgoing direction it is not send to the peer and in incoming direction it is not accepted and dropped.

When a route does not match the match rules of a route-mapping, no actions of that route-mapping are executed, specifically no properties are set, and the route is compared against the next route-mapping. If there is no further route-mapping or not a route-mapping to begin with the route is denied.

The name of a route-mapping is for description purposes and does not affect the functionality of the route-mapping itself. There can be even mappings with the same name if need be.

Mode: profile route-map

Step	Command	Purpose
1	node(pf-route-map)[name]# [no] route-map [<index>] (permit deny) <routemap-name>	Creates a new or deletes an existing route-mapping statement within the profile.

Match rules

When there is a route-mapping with no specified match-rule, it matches to all routes and the according action is executed. If there are multiple match-rules within the same route-mapping it depends on the type of match-rules: match-rules of the same type, more specifically multiple match ip-address rules, are a logically OR. This means only one of them must match to fulfill the match requirement. But match rules of a different type are logically AND. This means of each match-rule type at least one of these must match to fulfill the match requirement of the route-mapping.

The match ip-address command is the only match command where multiple possible matches can be specified, with multiple commands. There are three variants for matching a route with the match ip-address command:

- **all-subnets:** This is matched if the network is of the same size or smaller than the network specified by the rule and the network matches the network part from the rule which is not masked.
- **only-this-subnet:** This is matched if the network is of the exact same size as the network specified by the rule and the network matches the network part from the rule which is not masked.
- **subnet-range:** This is matched if the network is within the exact range of mask specified by the rule and the network matches the network part from the rule which is not masked.
 - **ge:** the size of the route mask must be smaller or the same size as specified.
 - **le:** the size of the route mask must be bigger or the same size as specified.

Note These options need to respect the condition **net mask <= ge <= le** to work properly.

Mode: profile route-map / route-map

Step	Command	Purpose
1	node(route-map)[name]# match ip-address <index> all-subnets <a.b.c.d/m>	Adds a new match ip-address all-subnets statement.

Step	Command	Purpose
2	node(route-map)[name]# match ip-address <index> only-this-subnet	Adds a new match ip-address only-this-subnet statement.
3	node(route-map)[name]# match ip-address <index> subnet-range <a.b.c.d/m> [ge <len>] [le <len>]	Adds a new match ip-address subnet-range statement.

It is not allowed have match ip-address *all-subnets* mixed with *only-this-subnet* or *subnet-range* commands together in the same route mapping.

Mode: profile route-map / route-map

Step	Command	Purpose
1	node(route-map)[name]# no match ip-address <index>	Removes a match ip-address statement.

It is allowed have the following match commands together with the match ip-address commands. In that case at least one match rule of each type must match to fulfill the match requirements of the route-mapping.

Mode: profile route-map / route-map

Step	Command	Purpose
1	node(route-map)[name]# [no] match ip-next-hop <a.b.c.d>	Configures a match for the next-hop ip-address of the route. Default: disabled.
2	node(route-map)[name]# [no] match metric <number>	Configures a match for the metric of the route. Default: disabled.

Set rules

The set rules are only relevant within a route-mapping with a permit rule. Because they are only executed when the route matched the route-mapping and the route-mapping allows to forward the route. If a route is permitted, then all the parameters specified from the set statements of the according route mapping are set on that route.

Mode: profile route-map / route-map

Step	Command	Purpose
1	node(route-map)[name]# [no] set ip-next-hop <a.b.c.d>	Configures to overwrite the next-hop ip-address of the route. This can be applied for incoming or outgoing route-mappings. Default: disabled.
2	node(route-map)[name]# [no] set metric <number>	Configures to overwrite the metric of the route. This can be applied for incoming or outgoing route-mappings. Default: disabled.

Step	Command	Purpose
3	node(route-map)[name]# [no] set weight <number>	Configures to overwrite the weight of the route. This can be applied only for incoming route-mappings. For outgoing route-mappings it has no effect. Default: disabled.
4	node(route-map)[name]# [no] set local-preference <number>	Configures to overwrite the local-preference of the route. This can be applied only for incoming route-mappings. For outgoing route-mappings it has no effect. Default: disabled.

Examples

This is an example of the BGP and route-mapping part of a configuration. It is showing how the route-mapping can be used in incoming direction for prioritizing routes from one neighbor over the routes received from the other neighbor.

```

profile route-map PREF_800
  route-map 1 permit ALL
  set local-preference 800

profile route-map PREF_900
  route-map 1 permit ALL
  set local-preference 900

context ip ROUTER
  bgp
    router-id 10.172.40.9
    as-identifier 100
    timers keepalive 10 holdtime 30
    no shutdown

    network 10.178.32.0/24

    neighbor 10.175.50.8
      remote-as 200
      use profile route-map PREF_800 in
      no shutdown

    neighbor 10.177.144.70
      remote-as 300
      ebgp-multihop 5
      use profile route-map PREF_900 in
      no shutdown

```

The second example shows how the route-mappings are used to filter routes in outgoing direction. Here we have three networks to announce over BGP:

- 78.144.16.5 exclusive for one neighbor

- 78.144.18.7 exclusive for the other neighbor
- 55.55.0.0 for both neighbors

The profiles filter the routes, which are not to be announced to the corresponding neighbors.

```

profile route-map MAP_VOICE
  route-map 1 permit RTM_VOICE
    match ip-address 1 only-this-subnet 55.55.0.0/16
    match ip-address 2 only-this-subnet 78.144.16.5/32

profile route-map MAP_DATA
  route-map 1 permit RTM_DATA
    match ip-address 1 only-this-subnet 55.55.0.0/16
    match ip-address 2 only-this-subnet 78.144.18.7/32

context ip ROUTER
  bgp
    router-id 10.172.40.9
    as-identifier 100
    timers keepalive 10 holdtime 30

    network 55.55.0.0/16
    network 78.144.16.5/32
    network 78.144.18.7/32
    no shutdown

    neighbor 10.175.50.8
      remote-as 200
      use profile route-map MAP_VOICE out
      no shutdown

    neighbor 10.177.144.70
      remote-as 300
      use profile route-map MAP_DATA out
      no shutdown

```

Troubleshooting

The following commands are useful to examine the configuration and the status of BGP.

Mode: operator

Command	Purpose
node # show bgp route	Shows all the routes BGP is seeing and which would be distributed to peers when not filtered out.
node # show bgp scan	Shows the result of BGP scanning for routes.
node # show bgp summary	Shows the neighbors and the connection status between this router and the neighbor.

Command	Purpose
node # show bgp neighbor [<peer>]	Shows detailed information about the neighbors.
node # restart bgp neighbor [<peer>]	Triggers an immediate exchange of all routes between this router and the neighbor. To make sure a route exchange happens after a change in route mappings for example.
node # restart bgp router	Resets the complete router. To get rid of not actual routes originated from this router and to make sure all recent configuration changes are applied on the connections between this router and its neighbors.

It is possible to trace run-time information on BGP.

Mode: operator

Command	Purpose
node # [no] trace bgp-mgmt	Traces configuration changes and supervision of bgp daemon.
node # [no] trace bgp-event	Traces bgp daemon interactions with system and neighbors.
node # [no] trace all-bgp	A collection of bgp-mgmt and bgp-event.

Chapter 28 **Load Balancing and Failover Configuration**

Chapter contents

Introduction	358
Policy Routing	359
Heartbeat Profile	359
Load Balancing Service	359
Load Balancing and Failover Configuration Task List	359
Create a Heartbeat Profile (Optional)	360
Start a Heartbeat (Optional)	361
Configure the Routing Table(s) for the DHCP or Cell-modem Default Route (Optional)	361
Create a Load Balancer Service	361
Assign Traffic to a Load Balancer Service	363
Display the Heartbeat Status (Optional)	364
Display the Load Balancer Service Status (Optional)	364
Examples	365
Load Balancing and Failover Example	365
Troubleshooting.....	367
I lose management connectivity to my device when all my WAN links are down	367
After all my WAN links go down, they never come back up	367

Introduction

Load balancing means distributing network traffic across multiple interfaces. Failover means detecting when an interface's path is down and redirecting traffic to the remaining interfaces. Consider the following scenario in which we would want to use the load balancing and failover feature:

Suppose we have three paths to the Internet:

- A DSL provider who provides us 23 Mbps upstream
- A cable provider who provides us 10 Mbps upstream
- A cell-modem provider

Suppose the cell-modem provider charges us more depending on how much data we send, so we only want to use it if the DSL and cable providers are both down. We want to load balance over the DSL and cable providers, and if they both go down, we fallback to the cell-modem provider (see [figure 45](#) on page 358).

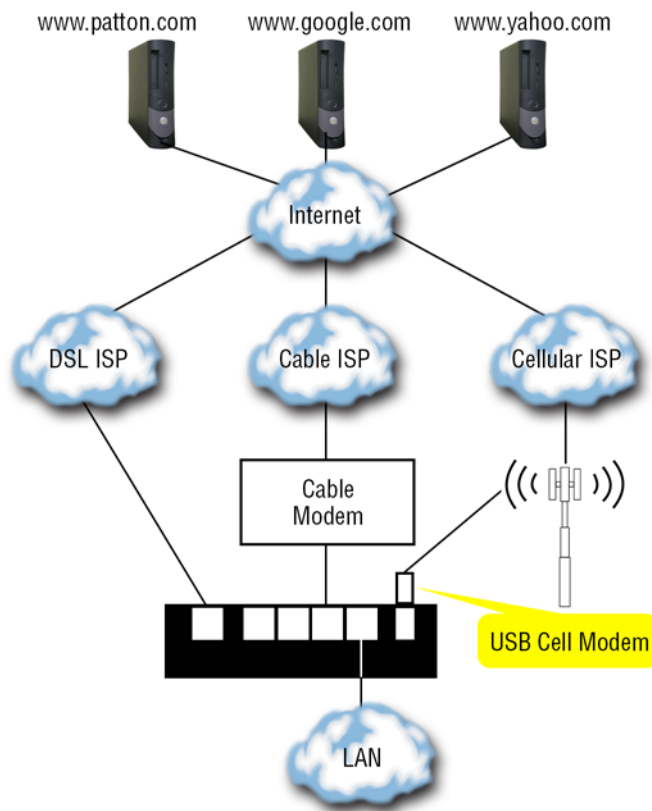


Figure 45. Load balancing and failover example

In this scenario, if both the DSL and cable providers are up, we have a total aggregate bandwidth of $23 + 10 = 33$ Mbps.

Note Load balancing is done on a connection-by-connection basis, so that any single connection can only achieve the maximum bandwidth of any single path, but if we have multiple users, each with multiple connections to the internet,

we can use the whole of the aggregate bandwidth. A connection is defined as a unique source and destination IP address and port (for TCP and UDP). For example, two PCs on the LAN accessing the same webserver on the Internet count as two different connections because the source IP address differs.

Load balancing and failover is implemented using the following elements:

- Policy Routing
- Heartbeat
- Load Balancing Service

Policy Routing

The first element in load balancing and failover is policy routing. We need multiple routing tables, each with a different path to the WAN. That is, each routing table has a default route that directs traffic out a different physical interface.

Heartbeat Profile

The second element in load balancing and failover is the heartbeat. A heartbeat consists of periodically pinging hosts on the WAN. It routes those pings using a given routing table. That routing table will route those pings out its default route. If the path to the hosts is down, whether it a cable is not plugged into the device, or whether it is an issue further away in the network, the host will fail to reply to the ping, and the heartbeat will detect that the route is down.

Load Balancing Service

The final element in load balancing and failover is the load balancing service. The load balancer service consists of a list of rules. Each rule specifies:

- A routing table
- A weight which is used to determine the percentage of connections to be routed by the table
- An optional heartbeat which monitors the table's route to the WAN and deactivates the rule when it is down

In addition, the load balancer service may have a fallback routing table which will be used to route all traffic when all of the rules are deactivated. If no fallback table is configured, it will drop all traffic when all of the rules are deactivated.

Supposing each routing table has a different default route that routes traffic destined to the WAN out a different interface, then the traffic to the WAN will be balanced across the multiple interfaces.

Load Balancing and Failover Configuration Task List

To configure load balancing and failover, perform the tasks described in the following sections:

- Create a heartbeat profile (optional) (page 360)
- Start a heartbeat (optional) (page 361)
- Configure the routing table(s) for the DHCP or cell-modem default route (optional) (page 361)

- Create a load balancer service (page 361)
- Assign traffic to a load balancer service (page 363)
- Display the heartbeat status (optional) (page 364)
- Display the load balancer service status (optional) (page 364)

Create a Heartbeat Profile (Optional)

A heartbeat profile defines the parameters of a heartbeat. A heartbeat is used to detect when a routing table's path to the WAN is down. The load balancer service requires a heartbeat in order to failover when a network goes down. If failover is not desired, you may skip this step.

Mode: Configure

Step	Command	Purpose
1	device(cfg)#profile heartbeat <i>pf-name</i>	Creates the heartbeat profile <i>pf-name</i> .
2	device(pf-hbt)[<i>pf-name</i>]#host <i>hostname</i>	It is recommended to use an IP address instead of a hostname because this avoids the need for a DNS lookup which introduces a single point of failure, i.e. if the DNS server goes down, all hosts will be unreachable. However, if the host's address is dynamic, a hostname is required. In this case, it is recommended to have at least two DNS servers configured. In addition, these DNS servers should be external. A DNS server on the LAN will not be able to reach its upstream server when all WAN links are down.
3 (optional)	device(pf-hbt)[<i>pf-name</i>]#interval <i>seconds</i>	Configures how often to ping each host. The default is 10 seconds.
4 (optional)	device(pf-hbt)[<i>pf-name</i>] #response-timeout <i>seconds</i>	Configures how long to wait for each host to reply to a ping. The default is 2 seconds.
5 (optional)	device(pf-hbt)[<i>pf-name</i>]#failure-count <i>count</i>	Configures how many times a host may fail to reply to a ping before that host is considered unreachable. The default is 3 times.
6 (optional)	device(pf-hbt)[<i>pf-name</i>]#require { all all-but at-least } <i>count</i>	Configures how many hosts may be unreachable before the network is considered down. The default is all hosts.

Start a Heartbeat (Optional)

Once the heartbeat profile is created, you can start a heartbeat to monitor a routing table. It will ping the hosts configured in the heartbeat profile using the routes configured in the routing table. The heartbeat detects whether the routing table's path to the WAN is up.

The load balancer service automatically performs this step, so you may skip it

Mode: Context IP

Step	Command	Purpose
1	device(ctx-ip)[ctx-name]#routing-table table-name	Enters the configuration mode for routing table <i>table-name</i> .
2	device(rt-tab)[ctx-name.table-name]#use profile heartbeat pf-name	Starts a heartbeat on routing-table <i>table-name</i> using the parameters configured in heartbeat profile <i>pf-name</i> .

Configure the Routing Table(s) for the DHCP or Cell-modem Default Route (Optional)

By default, the device adds the default route obtained by a DHCP or cell-modem IP address to all routing tables. However, the load balancer service works by assigning connections to different routing tables, each of which has a different route to the WAN. Therefore, we want to control which table(s) the device adds the default route to.

This step is only required if your WAN IP address is a DHCP or a cell-modem address. It is not required for static or IPCP addresses, because they do not obtain a default route.

Mode: IP Interface

Step	Command	Purpose
1	device(if-ip)[if-name]#accept default-route routing-table table-name	Adds routing table <i>table-name</i> to the list of tables to which the device will add the default route obtained by a DHCP or cell-modem address. If the list of tables is empty, the device will add the default route to all routing tables. The default is an empty list.

Create a Load Balancer Service

The load balancer service assigns each connection to one of a list of routing tables. Each routing table is expected to have a different route to the WAN, resulting in traffic being balanced over different interfaces.

The load balancer service consists of a list of rules. Each rule specifies:

- A routing table
- A weight which is used to determine the percentage of connections to be routed by the table
- An optional heartbeat which monitors the table's route to the WAN and deactivates the rule when it is down

In addition, the load balancer service may have a fallback routing table which will be used to route all traffic when all of the rules are deactivated. If no fallback table is configured, it will drop all traffic when all of the rules are deactivated.

Mode: IP Interface

Step	Command	Purpose
1	device(ctx-ip)[ctx-name]#service load-balancer svc-name	Creates the load balancer service <i>svc-name</i> .
2	device(svc-bal)[svc-name]#route weight weight dest-table table-name [heartbeat pf-name]	Adds a rule to the load balancer service. It will route a percentage of traffic determined by <i>weight</i> by routing table <i>table-name</i> , and optionally use heartbeat <i>pf-name</i> to monitor the table and deactivate the rule when it is down. This command will automatically start heartbeat <i>pf-name</i> on routing table <i>table-name</i> if it is not started already.
3 (optional)	device(svc-bal)[svc-name]#route fallback dest-table table-name	Configures <i>table-name</i> as the fallback routing table. The default is no fallback routing table.

Example: Determining percentages for load balancer service.

The following example explains how to determine what percentage of connections will be assigned to each routing table based on weights and heartbeat status.

```
context ip ROUTER

service load-balancer WAN
rule 1 weight 9 dest-table WAN1 heartbeat HB
rule 1 weight 7 dest-table WAN2 heartbeat HB
rule 1 weight 3 dest-table WAN3 heartbeat HB
rule 1 weight 1 dest-table WAN4 heartbeat HB
```

The percentage of connections assigned to each routing table is based on the rule's weight divided by the weight of all the other active rules.

First suppose the heartbeat is alive for all the rules. Then the total weight is $9 + 7 + 3 + 1 = 20$, and the percentage of connections assigned to each routing table is:

- WAN1: $(9 / 20) * 100 = 45\%$
- WAN2: $(7 / 20) * 100 = 35\%$
- WAN3: $(3 / 20) * 100 = 15\%$
- WAN4: $(1 / 20) * 100 = 5\%$

Now suppose the heartbeat for WAN2 and the heartbeat for WAN3 detect that the table's route to the WAN is down and deactivate these rules. Now the total weight is $9 + 1 = 10$, and the percentage of connections assigned to each routing table is:

- WAN1: $(9 / 10) * 100 = 90\%$
- WAN2: deactivated, 0%
- WAN3: deactivated, 0%
- WAN4: $(1 / 10) * 100 = 10\%$

Assign Traffic to a Load Balancer Service

The load balancer service assigns each connection to one of a list of routing tables. Each routing table is expected to have a different route to the WAN, resulting in traffic being balanced over different interfaces.

The load balancer service consists of a list of rules. Each rule specifies:

- A routing table
- A weight which is used to determine the percentage of connections to be routed by the table
- An optional heartbeat which monitors the table's route to the WAN and deactivates the rule when it is down

In addition, the load balancer service may have a fallback routing table which will be used to route all traffic when all of the rules are deactivated. If no fallback table is configured, it will drop all traffic when all of the rules are deactivated.

Mode: IP Interface

Step	Command	Purpose
1	device(if-ip)[if-name]#route [packet-match] dest-service svc-name	Sends traffic received by IP interface <i>if-name</i> to load balancer service <i>svc-name</i> . Optionally, a <i>packet-match</i> expression may be supplied so that only packets matching the expression are sent to the load balancer service. See Chapter 40, "Packet Matching" on page 458 for details on this expression.

Optionally, we may also load balance locally generated traffic.

Mode: Context IP

Step	Command	Purpose
1	device(ctx-ip)[ctx-name]#local	Enters the configuration mode for locally generated traffic.

Step	Command	Purpose
2	device(local)[ctx-name]#route [packet-match] dest-service svc-name	Sends locally generated traffic to load balancer service <i>svc-name</i> . Optionally, a <i>packet-match</i> expression may be supplied so that only packets matching the expression are sent to the load balancer service. See Chapter 40, "Packet Matching" on page 458 for details on this expression.

Display the Heartbeat Status (Optional)

A command exists to display the heartbeat status.

Mode: Operator execution

Step	Command	Purpose
1	device>show profile heartbeat pf-name	Displays the configuration and status of heartbeat <i>pf-name</i>

Example: Displaying heartbeat status.

The following example shows that the heartbeat status. We see that both the DSL and cable providers are up.

```
device>show profile heartbeat HB_WAN
Heartbeat Profile: HB_WAN
=====

Hosts
-----

www.patton.com
www.google.com
www.yahoo.com

Interval:                10 seconds
Failure Count:           3
Response Timeout:        2 seconds
Require:                  all hosts

Heartbeat      Alive  Host                Alive  Address              Failure Count
-----
ROUTER.RT_DSL.A  true  www.patton.com     true   40.130.246.195      0
                true  www.google.com     true   172.217.1.4         0
                true  www.yahoo.com      true   98.139.183.24       0
ROUTER.RT_CABLE.A true  www.patton.com     true   40.130.246.195      0
                true  www.google.com     true   172.217.1.4         0
                true  www.yahoo.com      true   98.139.183.24       0
```

Display the Load Balancer Service Status (Optional)

A command exists to display the load balancer service status.

Mode: Operator execution

Step	Command	Purpose
1	device>show service load-balancer <i>ctx-name svc-name</i>	Displays the configuration and status of load balancer service <i>svc-name</i> in <i>ctx-name</i> .

Example: Displaying load balancer service status.

The following example shows the load balancer service status. We see that both the DSL and cable providers are up, and we are forwarding to them. We also see that at some time both of them were down and we forwarded 10 packets to the cell-modem provider.

```
device>show service load-balancer ROUTER SVC_WAN
Load Balancer Service: SVC_WAN
=====

Index      Weight  Table           Heartbeat  Alive  Connections  Packets  Bytes
-----
-----
1          23     RT_DSL          HB_WAN    true   52981         52981   11649314
2          10     RT_CABLE        HB_WAN    true   26540         26540   5835495
<fallback> RT_CELL-MODEM              10       2205
<drop>                0         0
```

Examples

Load Balancing and Failover Example

The following example configures load balancing and failover for the application described in the Introduction to this chapter.

```
# We will ping www.patton.com, www.google.com, and www.yahoo.com to determine if
# a given provider is up.
profile heartbeat HB_WAN
  host www.patton.com
  host www.google.com
  host www.yahoo.com

profile napt MASQ

context ip ROUTER

# We have three paths to the Internet: 1) a DSL provider who provides us
# 23 Mbps upstream, 2) a cable provider who provides us 10 Mbps upstream, and
# 3) a cell-modem provider. The cell-modem provider charges us more depending
# on how much data we send, so we only want to use it if the DSL and cable
# providers are both down. We load balance over the DSL and cable providers,
# and if they both go down, we fallback to the cell-modem provider.
service load-balancer SVC_WAN
  route 1 weight 23 dest-table RT_DSL heartbeat HB_WAN
  route 2 weight 10 dest-table RT_CABLE heartbeat HB_WAN
  route fallback dest-table RT_CELL-MODEM
```

```
# Load balance traffic generated by this device.
local
  route 1 dest-service SVC_WAN

# Load balance traffic routed through this device from the LAN to the WAN.
interface IF_LAN
  ipaddress IF_LAN 192.168.1.1/24
  route 1 dest-service SVC_WAN

# The DSL, cable, and cell-modem providers all supply us with a DHCP address
# and a default route. We add the default route that each one supplies to a
# different table. Any traffic destined for the WAN that the load balancer
# assigns to one of these tables will be forwarded out the appropriate
# interface. Each of the providers expect us to NAT all traffic we send to
# them.
interface IF_DSL
  ipaddress ADDR_DSL dhcp
  no accept hostname
  accept default-route routing-table RT_DSL
  use profile napt MASQ ADDR_DSL

interface IF_CABLE
  ipaddress ADDR_CABLE dhcp
  no accept hostname
  accept default-route routing-table RT_CABLE
  use profile napt MASQ ADDR_CABLE

interface IF_CELL-MODEM
  ipaddress ADDR_CELL-MODEM cellmodem
  no accept hostname
  accept default-route routing-table RT_CELL-MODEM
  use profile napt MASQ ADDR_CELL-MODEM

# Ethernet 0 0 is connected to the LAN.
port ethernet 0 0
  bind interface ROUTER IF_LAN
  no shutdown

# Ethernet 0 1 is connected to a cable modem, which is connected to the cable
# provider.
port ethernet 0 1
  bind interface ROUTER IF_CABLE
  no shutdown

# DSL 0 0 is connected to the DSL provider.
port dsl 0 0
  mode cpe
  service-mode 8-wire
  bind interface ROUTER IF_DSL
  no shutdown

# The USB port is connected to a cell-modem, which is connected to the cell-
# modem provider.
port virtual cell-modem 1
  link usb vendor 0x19d2 product 0x0157 priority 1
```

```
apn fast.t-mobile.com
bind interface ROUTER IF_CELL-MODEM
no shutdown
```

Troubleshooting

I lose management connectivity to my device when all my WAN links are down

If your load balancer service does not have a fallback route configured, it is possible to lose management connectivity to your device, even if you are managing it from the LAN.

You may have a configuration similar to the following:

```
profile heartbeat HB_WAN
  www.patton.com

context ip ROUTER

  service load-balancer SVC_WAN
    route weight 1 dest-table RT_WAN1 heartbeat HB_WAN
    route weight 1 dest-table RT_WAN2 heartbeat HB_WAN
    no route fallback

  local
    # route dest-ip 192.168.1.0/24 dest-table DEFAULT
    route dest-service SVC_WAN

  interface LAN
    ipaddress LAN 192.168.1.1/24
    # route dest-ip 192.168.1.1 dest-table DEFAULT
    route dest-service SVC_WAN
```

In this case, when both the WAN links are down, the load balancer service drops all packets that are sent to it because it has no fallback route. This means that when your management traffic (e.g. SSH or web traffic to the device itself) enters the LAN interface and is sent to the load balancer service, it is dropped. Likewise, when the device responds to you with locally generated management traffic, it is sent to the load balancer service and dropped.

To prevent yourself from losing management connectivity to your device from the LAN, add the two lines that are commented out in the configuration above.

After all my WAN links go down, they never come back up

This can happen if your heartbeat specifies hosts by hostname instead of by IP address, and if your DNS server is on the LAN, as shown in the [figure 46](#) on page 368.

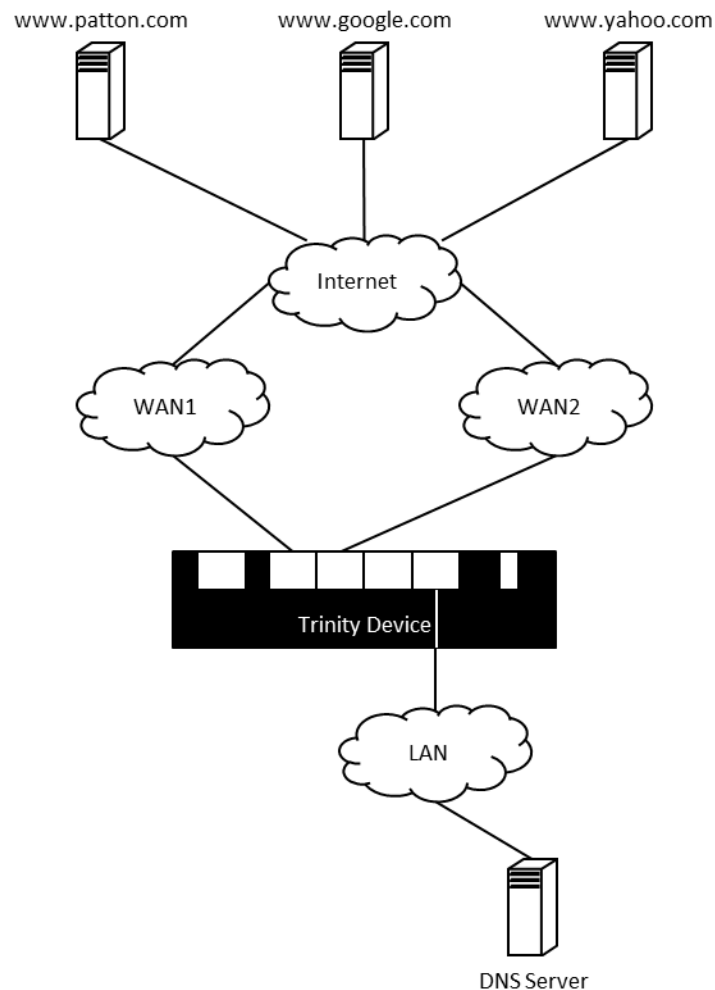


Figure 46. After all my WAN links go down, they never come back up

In this case, what happens when the heartbeat for both WAN1 and WAN2 is down is:

1. The Trinity device's heartbeat continues to attempt to ping the hosts (e.g. `www.patton.com`, `www.google.com`, and `www.yahoo.com`) to detect when WAN1 and WAN2 come up.
2. To do so, the Trinity Device queries the DNS server for their IP addresses.
3. The DNS server will cache those addresses for a period of time, but after that time expires, it will request the IP addresses from an upstream DNS server on the Internet.
4. The Trinity Device will drop the DNS server's request to the upstream server because the heartbeat has detected that neither WAN1 nor WAN2 is up.
5. The DNS server will respond to the Trinity Device that it does not know the IP addresses for the hosts.
6. The Trinity Device will not be able to ping the hosts because it does not know their IP addresses, so the heartbeat will remain down.

These steps will repeat indefinitely.

Chapter 29 **Fast-Path**

Chapter contents

Introduction.....	370
Fast-Path Configuration.....	370

Introduction

Fast Path is a new feature to speed-up the routing performance of Trinity devices. Fast Path only routes the first packet of a UDP/TCP flow and learns to which interface it is routed and what packet header manipulations are being performed (e.g. by ACL, NAT, etc.). All subsequent packets of the same layer 4 flow bypass the router and take the fast path.

This is currently an experimental feature and therefore, the routing fast path is disabled by default. If you experience routing performance limitations you can switch it on with the CLI commands described below.

Note Currently, the fast path feature only works for UDP and TCP flows over IPv4. All other protocols (IPv6 or other transport protocols) always take the (slower) routing path.

Fast-Path Configuration

To enable/disable the routing fast path, enter the following command(s):

Mode: configure

Step	Command	Purpose
1	node(cfg)# context ip ROUTER	Enter the IP context configuration mode.
2	node(ctx-ip)[Device]# [no] fast-path	Enables/Disables? fast-path processing for the IPv4 router.

Optionally—usually only when requested by our technical supporters—you may tweak the routing fast path by entering one or multiple of the following commands:

Mode: configure

Step	Command	Purpose
1	node(cfg)# context ip ROUTER	Enter the IP context configuration mode.
2	node(ctx-ip)[Device]# fast-path	Enter the configuration mode of the routing fast path.
3	node(fp)[Device]# flow idle-lifetime [<interval>]	Enter the time interval in ms after which idle UDP/TCP flows are considered closed. Since only 1024 UDP and 1024 TCP can be tracked simultaneously by the fast path, this interval should be short. On the other hand, it must be long enough to keep UDP flows that are rarely sending packets open (TCP flows are closed when routing the FIN packet). The default idle-life-time interval is 5 minutes (300000 ms).

Step	Command	Purpose
4	node(fp)[Device]# flow route-interval [<interval>]	Enter the time interval in ms after which a packet of a UDP/TCP connection again passes the router too keep the stateful firewall and NAPT happy. The default route interval is 30 seconds (30000 ms).
5	node(fp)[Device]# garbage-collection interval [<interval>]	Enter the time interval in ms after which the fast path destroys idle flows and makes memory available for new connections. The default value is 10 seconds (10000 ms).

Chapter 30 NAT/NAPT Configuration

Chapter contents

Introduction	373
Dynamic NAPT	373
Static NAPT	374
Dynamic NAT	375
Static NAT	375
NAPT traversal	376
NAT/NAPT Configuration Task List.....	376
Creating a NAPT Profile	376
Configuring a NAPT DMZ host	377
Activate NAT/NAPT	378
Displaying NAT/NAPT Configuration Information	378

Introduction

This chapter provides a general overview of Network Address (Port) Translation and describes the tasks involved in its configuration.

For further information about the functionality of Network Address Translation (NAT) and Network Address Port Translation (NAPT), consult the RFCs 1631 and 3022. This chapter applies the terminology defined in RFC 2663.

Trinity provides four types of NAT/NAPT:

- Dynamic NAPT (Cisco terminology: NAT Overload)
- Static NAPT (Cisco terminology: Port Static NAT)
- Dynamic NAT
- Static NAT

You can combine these types of NAT/NAPT without any restriction. One type of profile, the 'NAPT Profile', holds the configuration information for all four types where configuration is required. The remainder of this Section shortly explains the behavior of the different NAT/NAPT types.

Dynamic NAPT

Dynamic NAPT is the default behavior of the NAT/NAPT component. It allows hosts on the local network to access any host on the global network by using the global interface address as source address. It modifies not only the source address, but also the source port, so that it can tell different connections apart (NAPT source ports are in the range 8,000 to 16,000). UDP and TCP connections from the local to the global network trigger the creation of a dynamic NAPT entry for the reverse path. If a connection is idle for some time (UDP: 2 minutes, TCP: 12 hours) or gets closed (only TCP), the dynamic NAPT entry is removed.

An enhancement of the Dynamic NAPT allows to define subsets of hosts on the local network that shall use different global addresses. Up to 20 subsets with their respective global addresses are possible. Such a global NAPT address can be any IP address as long as the global network routes the traffic to the global interface of the NAT/NAPT component.

Figure 47 illustrates the basic and enhanced behavior of the Dynamic NAPT. The big arrows indicate the direction of the connection establishment. Although only a local host can establish a connection, traffic always flows in both directions.

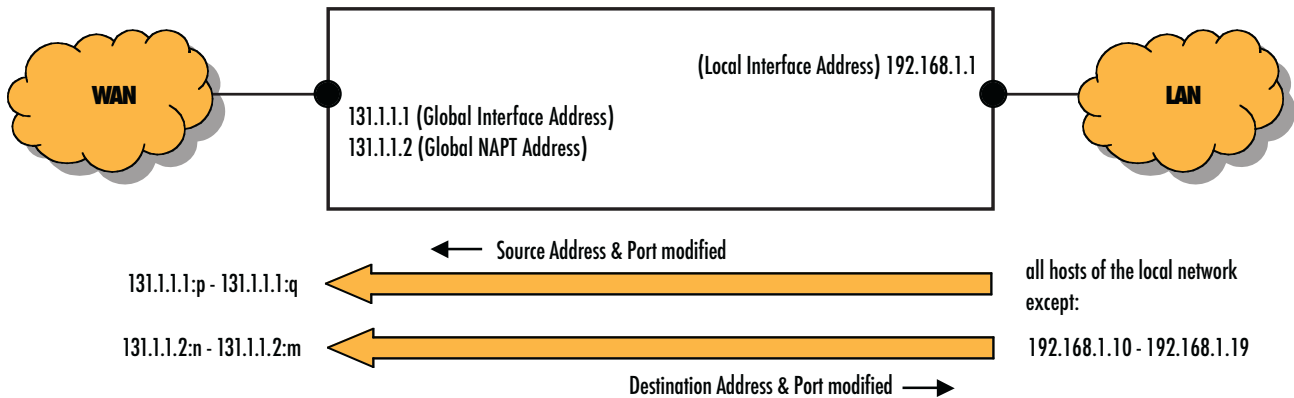


Figure 47. Dynamic NAPT

Static NAPT

Static NAPT makes selected services (i.e. ports) of local hosts globally accessible. Static NAPT entries map global addresses/ports to local addresses/ports. The global address can either be the address of the global interface or a configured global NAPT address. Usually, the local and the global port of a static NAPT entry are the same; however, they may be different.

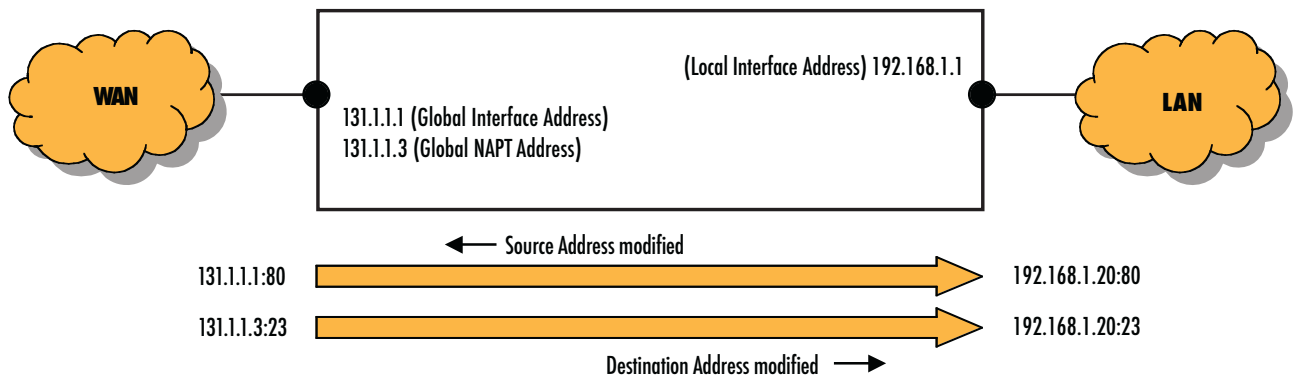


Figure 48. Static NAPT

Note Be careful when mapping ports the Patton device uses itself (e.g. Telnet, TFTP) because the device might become inaccessible.

Dynamic NAT

NAT only modifies addresses but not ports. Dynamic NAT assigns a global address from a global NAT address pool each time a local host wants to access the global network. It creates a dynamic NAT entry for the reverse path. If a connection is idle for some time (2 minutes), the dynamic NAT entry is removed. Should Dynamic NAT run out of global addresses, it lets Dynamic NAPT handle the connection (which may lead to an unexpected behavior).

Dynamic NAT is particularly useful for protocols that do not build on UDP or TCP but directly on IP (e.g. GRE, ESP). See also section “[NAPT traversal](#)” on page 376.

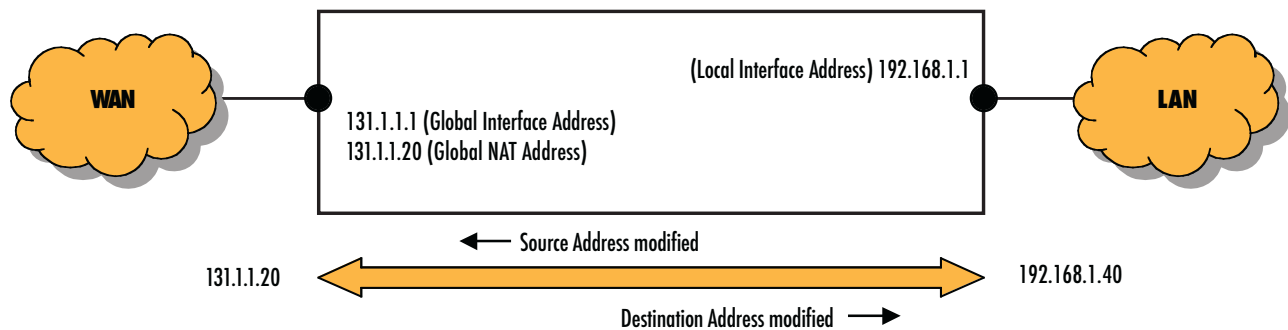


Figure 49. Dynamic NAT

Static NAT

Static NAT makes local hosts globally accessible. Static NAT entries map global addresses to local addresses. The global address must be a configured global NAT address. It cannot be the address of the global interface since this would break connectivity to the Patton device itself.

Static NAT is particularly useful for protocols that do not build on UDP or TCP but directly on IP (e.g. GRE, ESP). See also section “[NAPT traversal](#)” on page 376.

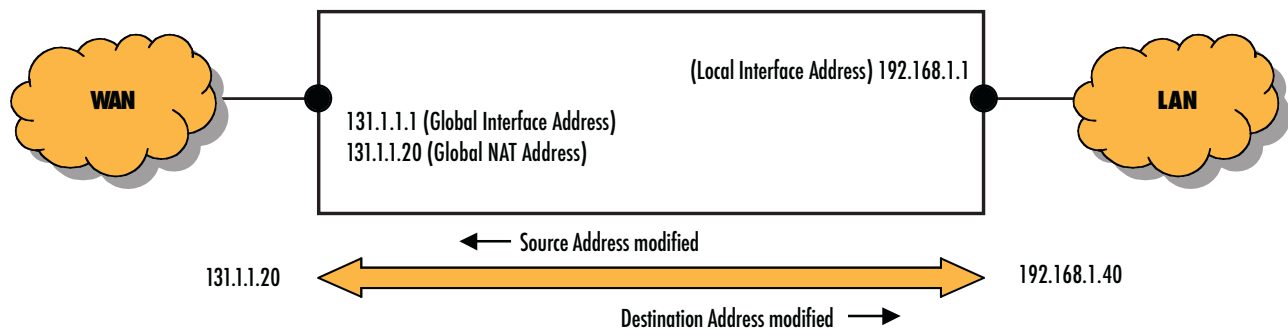


Figure 50. Static NAT

NAPT traversal

Protocols that do not build on UDP or TCP but directly on IP (e.g. GRE, ESP), and protocols that open additional connections unknown to the NAT/NAPT component (e.g. FTP, SIP), do not easily traverse a NAPT.

The Trinity NAPT can handle one GRE (Generic Routing Encapsulation) connection and one ESP (Encapsulating Security Payload) connection at a time. It also routes ICMP messages back to the source of the concerned connection or to the source of an ICMP Ping message.

To enable NAPT traversal of protocols that open additional connections, the NAPT component must analyze these protocols at the Application Level in order to understand which NAPT entries for additional connections it should create and which IP addresses/ports it must modify (e.g. for voice connections in addition to signaling connections). It performs this task for the protocol FTP. Other protocols such as SIP cannot traverse the Trinity NAPT.

NAT/NAPT Configuration Task List

To configure the NAT/NAPT component, perform the tasks in the following sections:

- Creating a NAPT profile (see page 376)
- Activating NAT/NAPT (see page 376)
- Displaying NAT/NAPT configuration information (see page 378)

Creating a NAPT Profile

A NAPT profile defines the behavior of the NAT/NAPT, comprising all four types of NAT/NAPT (this profile is called 'NAPT profile' and not 'NAT/NAPT profile for historical reasons). Several NAPT profiles are admissible but there is only one NAT/NAPT component.

Procedure: To create a NAPT profile and to configure the required types of NAT/NAPT

Mode: Configure

Step	Command	Purpose
1	device(cfg)#profile napt name	Creates the NAPT profile <i>name</i> and activates the basic behavior of the Dynamic NAPT
2 (optional)	device(pf-napt)device#range <i>local-ip-range-start local-ip-range-stop global-ip</i>	Configures and activates the enhanced behavior of the Dynamic NAPT: <i>local-ip-range-start</i> and <i>local-ip-range-stop</i> define the subset of local hosts that use the global NAT address <i>global-ip</i> to access to global network. (max. 20 entries) The IP ranges of different Dynamic NAPT entries must not overlap each other.
3 (optional)	node(pf-napt)[<name>]# static {udp tcp} <local-ip> <local-port> [<global-ip>] [<global-port>]	Creates a Static NAPT entry: <i>local-ip/local-port</i> is mapped to <i>global-ip/global-port</i> . If <i>global-port</i> is omitted, <i>local-port</i> is used on both sides. If <i>global-ip</i> is omitted, the global address is the address of the global interface. (<i>local-port</i> to be either a single port as before, or a range of ports, e.g. "5000..5999") (max. 20 UDP and 20 TCP entries)

Step	Command	Purpose
4 (optional)	device(pf-napt)device#range <i>local-ip-range-start local-ip-range-stop global-ip-start global-ip-stop</i>	Configures and activates the Dynamic NAT: <i>local-ip-range-start</i> and <i>local-ip-range-stop</i> define the subset of local hosts that use an address from the global NAT address pool to access to global network. <i>global-ip-start</i> and <i>global-ip-stop</i> define the global NAT address pool. (max. 20 entries) The IP ranges of different Dynamic NAT entries must not overlap each other.
5 (optional)	device(pf-napt)device#static <i>local-ip global-ip</i>	Creates a Static NAT entry: <i>local-ip</i> is mapped to <i>global-ip</i> . (max. 20 entries)
6 (optional)	device(pf-napt)device#static { ah esp gre ipv6 } <i>local_ip</i> <i>[global_ip]</i> .	Creates a static NAT entry: traffic of the IP protocol AH, ESP, GRE, or IPv6 respectively directed to the <i>global_ip</i> is forwarded to the <i>local_ip</i> .

Use no in front of the above commands to delete a specific entry or the whole profile.

Note The command `icmp default` is obsolete.

Example: Creating a NAPT Profile

The following example shows how to create a new NAPT profile *access* that contains all settings necessary to implement the examples in section “Introduction” on page 373.

```
device(cfg)#profile napt access
device(pf-napt)[access]#range 192.168.1.10 192.168.1.19 131.1.1.2
device(pf-napt)[access]#static tcp 192.168.1.20 80
device(pf-napt)[access]#static tcp 192.168.1.20 23 131.1.1.3
device(pf-napt)[access]#range 192.168.1.30 192.168.1.39 131.1.1.10 131.1.1.15
device(pf-napt)[access]#static 192.168.1.40 131.1.1.20
device(pf-napt)[access]static ah 192.168.1.41 131.1.1.120
```

Configuring a NAPT DMZ host

The NAPT allows a DMZ host to be configured, which receives any inbound traffic on the global NAPT interface, which:

- Is not translated by any static or dynamic NAPT entry and
- Is not handled by the device itself.

The following procedure shows how a DMZ host can be configured.

Mode: NAPT profile

Step	Command	Purpose
1	device(pf-napt)device#[no] dmz-host <i>dmz-host-ip-address [global-ip-address]</i>	Configures a DMZ host. The global-ip-address must only be specified, if the DMZ host shall handle the inbound traffic for a different NAPT global IP address than the gateways global interface IP address.

Activate NAT/NAPT

To enable NAPT on a WAN port, the user has to "use" a NAPT profile on the corresponding IP interface. In order to work, NAPT requires a global IP address. Since there now can be multiple IP addresses per IP interface, the user has to specify which of the IP interface addresses shall be used as NAPT global address.

Step	Command	Purpose
1	<code>node(cfg)#context ip [ROUTER].</code>	Enters the IP context mode for the default virtual router.
2	<code>node(ctx-ip)[ROUTER]#interface <name></code>	Enters the IP interface that is bound to the WAN port.
3	<code>node(ip-if)[ROUTER.<name>]#use profile napt <pfName> <label></code>	Enables NAPT on the WAN interface and uses its IP address with label <label> as global address.

The <pfName> parameter specifies the NAPT profile to be used for this IP interface. The <label> parameter specifies the name of the IP address on the same IP interface.

Note If the <label> parameter refers to a dynamic address (such as DHCP), the global NAPT address changes automatically when we got a new DHCP lease. If that DHCP address is released, all NAPT translations that require a global interface address are not active while translation rules that explicitly specify a global IP address still work.

Note You can use the same NAPT profile on multiple IP interfaces. In this case, the linked IP interface address will be used as global IP address on the corresponding port.

Example: Create an IP interface with one static and a DHCP address and bind the NAPT profile to the dynamic DHCP address.

```
node>enable
node#configure
node(cfg)#profile napt DYNAMICNAPT
node(cfg)#context ip ROUTER
node(ctx-ip)[ROUTER]#ip interface WAN
node(ip-if)[ROUTER.WAN]#ipaddress STATIC 10.1.1.1/24
node(ip-if)[ROUTER.WAN]#ipaddress DHCP
node(ip-if)[ROUTER.WAN]#use profile napt DYNAMICNAPT DHCP
node(ip-if)[ROUTER.WAN]#port ethernet 0 0
node(prt-eth)[0/0]#bind interface ROUTER WAN
node(prt-eth)[0/0]#no shutdown
```

Displaying NAT/NAPT Configuration Information

Two commands are available to display an existing NAPT profile. There is no command yet to display the dynamic entries of a NAT/NAPT component.

Procedure: To display NAT/NAPT configuration information

Mode: Configure

Step	Command	Purpose
1	device(cfg)#show profile NAPT	Displays the available NAPT profiles
2	device(cfg)#show profile NAPT <i>name</i>	Displays the NAPT profile <i>name</i>

Example: Display NAT/NAPT configuration information

```

device(pf-napt)[access]#show profile NAPT access
NAPT profile access:
-----
STATIC NAPT MAPPINGS
  Protocol      Local IP          Local Port      Global IP        Global Port
  -----      -
  tcp           192.168.1.20     80              0.0.0.0          80
  tcp           192.168.1.20     23              131.1.1.3        23

STATIC NAT PROTOCOL MAPPINGS
  Protocol Local IP          Global IP
  -----
  ah       192.168.1.41     131.1.1.120

STATIC NAT MAPPINGS
  Local IP      Global IP
  -----
  192.168.1.40  131.1.1.20

STATIC NAPT RANGE MAPPINGS
  Local IP Start  Local IP Stop  Global IP
  -----
  192.168.1.10   192.168.1.19  131.1.1.15

STATIC NAT RANGE MAPPINGS
  Local IP Start  Local IP Stop  Global IP Start  Global IP Stop
  -----
  192.168.1.30   192.168.1.39  131.1.1.10      131.1.1.15

```

Chapter 31 **DHCP Configuration**

Chapter contents

Introduction.....	381
DHCP-client Configuration Tasks.....	382
Configure an IP interface for DHCP	382
Release or Renew a DHCP Lease Manually (advanced)	383
Remove a DHCP address from an IP interface	383
Capture Debug Output from DHCP-client	383

Introduction

This chapter provides an overview of the Dynamic Host Configuration Control Protocol (DHCP) and describes the tasks involved in its configuration. This chapter includes the following sections:

- DHCP-client configuration tasks (see page 382)

The Dynamic Host Configuration Protocol (DHCP) automates the process of configuring new and existing devices on TCP/IP networks. DHCP performs many of the same functions a network administrator carries out when connecting a computer to a network. Replacing manual configuration by a program adds flexibility, mobility, and control to networked computer configurations.

The tedious and time-consuming method of assigning IP addresses was replaced by automatic distributing IP addresses. The days when a network administrator had to manually configure each new network device before it could be used on the network are in the past.

In addition to distributing IP addresses, DHCP enables configuration information to be distributed in the form of DHCP options. These options include, for example, the default router address, domain name server addresses, the name of a boot file to load etc.

A new expression in DHCP is lease. Rather than simply assigning each DHCP-client an IP address to keep until the client is done with it, the DHCP-server assigns the client an IP address with a lease; the client is allowed to use the IP address only for the duration of that lease. When the lease expires, the client is forced to stop using that IP address. To prevent a lease from expiring, which essentially shuts down all network access for the client, the client must renew its lease on its IP address from time to time.

The DHCP-server and DHCP-client are illustrated in [figure 51](#).

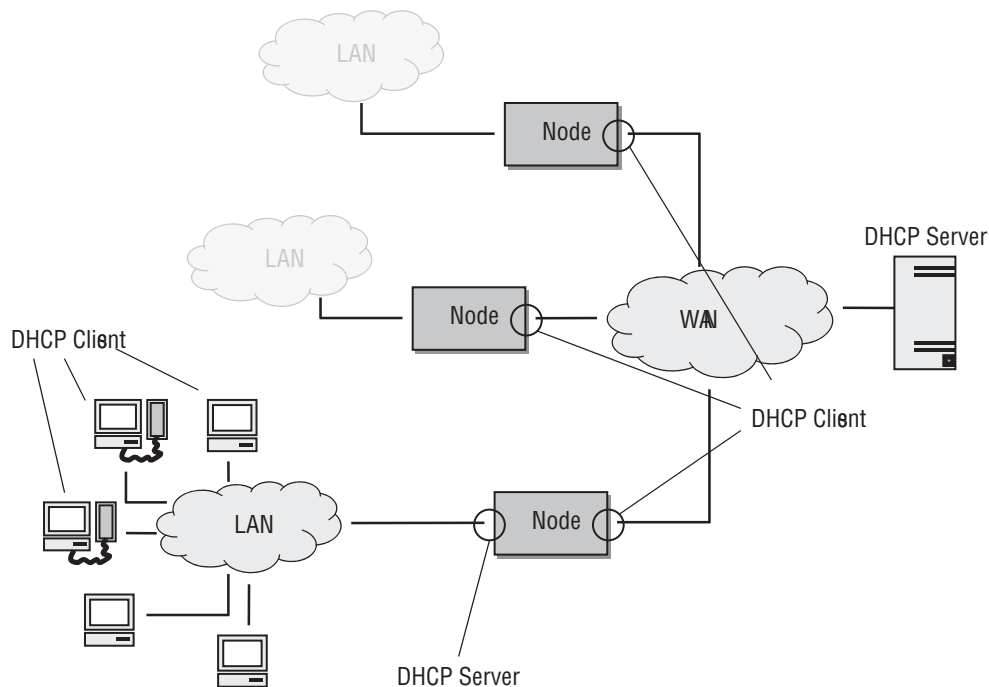


Figure 51. DHCP-client and DHCP-server

DHCP-client Configuration Tasks

To configure the device as DHCP-client, perform the steps mentioned below.

- Configure an IP interface for DHCP
- Release or renew a DHCP lease manually (advanced) (see page 383)
- Remove a DHCP address from an IP interface
- Capture debug output from the DHCP-client (see page 383)

Configure an IP interface for DHCP

On every created IP interface a DHCP-client could be enabled. If enabled, the device gets one IP address for this interface from a DHCP-server. Additionally, other configuration information is received for this IP interface, i.e. the default gateway, DNS server IP addresses, and the host name.

Note Next to a single DHCP address, an IP interface may foster an arbitrary number of static (manually-configured) IP addresses. The DHCP address distinguishes from the static addresses by its *label* set to *DHCP*.

To enable the DHCP-client on an IP Interface, configure the DHCP value with the option “ipaddress” by performing the steps described below.

Mode: configure

Step	Command	Purpose
1	<code>device(cfg)#context ip [ROUTER]</code>	Enters the IP context configuration mode for the default virtual router.
2	<code>device(ctx-ip)[ROUTER] #interface <name></code>	Enters the configuration mode of an existing IP interface or creates a new IP interface.
3	<code>device(if-ip)[<name>]#ipaddress [<label>] dhcp [<req-addr>] [route-metric] [ignore { route dns host }]</code>	If no address label is specified the name of the interface will be taken. Creates a DHCP IP address and requests a specific <req-addr> or any address from the DHCP server; potentially ignores information received from the server such as route (default gateway), dns (DNS server IP addresses), and host (host name). If the route-metric is omitted then that device will have a metric of 0 and become the default route.

Release or Renew a DHCP Lease Manually (advanced)

After enabling the DHCP-client, the interface receives a DHCP lease from the DHCP-server. To manually release and/or renew this DHCP lease use the command described below.

Mode: interface

Step	Command	Purpose
1	device(ip-if)[ROUTER name]#ipaddress DHCP release	Releases DHCP lease. (See note)
2	device(ip-if)[ROUTER name]#ipaddress DHCP renew	Gets a DHCP lease from the DHCP-server

Note If you are connected by Telnet or SSH over the IP interface on which you release the DHCP lease, the connection is lost after entering the command **ipaddress DHCP release**. You need another way (i.e. another static IP address or another IP interface) to connect to the device again and to enter the command **ipaddress DHCP renew**.

Remove a DHCP address from an IP interface

To completely disable the DHCP client on an IP interface, remove the IP address with the “no ipaddress” command.

Mode: interface

Step	Command	Purpose
1	device(ip-if)[ROUTER name]#[no] ipaddress DHCP	Disables the DHCP-client on the current IP interface.

Note If you are connected by Telnet or SSH over the IP address you delete, the connection is lost after entering the command **no ipaddress**. You need another way (e.g. another static IP address, another IP interface, or console access) to connect to the device again.

Capture Debug Output from DHCP-client

This procedure describes how to enable/disable the IP debug monitor, which shows traces of the DHCP client.

Mode: Any

Step	Command	Purpose
1	device>#[no] debug ip <detail> <detail level 0-5>	Enables/disables the DHCP-client debug monitor

Example: Tracing DHCP renew and release requests

This example shows how to enable the IP debug monitor and the debug output generated by the consecutive commands **ipaddress DHCP release** and **ipaddress DHCP renew** on IP interface *LAN*.

```
host (if-ip) [ROUTER.LAN] #debug ip
```

```

23:05:02 IP # [INF] [ROUTER.LAN.DHCP] State: up (normal) | Event: release
23:05:02 IP # [INF] [ROUTER.LAN.DHCP] Restart Mode: normal -> release
23:05:02 IP # [INF] [ROUTER.LAN.DHCP] State: up (release) | Action: release
dynamic DHCP address on dev eth0
23:05:02 IP # [DBG] [eth0] Releasing DHCP address
23:05:02 IP # [DBG] [ROUTER.LAN.DHCP] onUpdate: restartMode
23:05:02 IP # [INF] [eth0] DHCP Client >> Unicasting a release of 172.16.60.4 to
172.16.20.10
23:05:02 IP # [INF] Sending release...
23:05:02 IP # [INF] [eth0] DHCP Client >> deconfig: interface="eth0"
23:05:02 IP # [INF] [eth0] DHCP Client: Release DNS server 172.16.20.20
23:05:02 IP # [INF] [eth0] DHCP Client: Entering released state
23:05:02 IP # [INF] [eth0] DHCP Client: Release DNS server 172.16.20.10
23:05:02 IP # [INF] [eth0] DHCP Client: Release default route
23:05:03 IP # [INF] [eth0] DHCP Client: Release IP address
23:05:03 IP # [INF] [ROUTER.LAN.DHCP] State: up (release) | Event: dynamic DHCP
address released, reason=DHCP RELEASED
23:05:03 IP # [INF] [ROUTER.LAN.DHCP] State: up (release) | Action: revoke
address 172.16.60.4/19 from dev eth0
23:05:03 IP # [INF] [eth0] Kernel << ip addr del 172.16.60.4/19 dev eth0
23:05:03 IP # [INF] [ROUTER.LAN.DHCP] State: up (release) -> revoking
23:05:03 IP # [INF] [ROUTER.LAN.DHCP] State: up (release) | Action: decrement
active address counter
23:05:03 IP # [DBG] [ROUTER.LAN.DHCP] onUpdate: rcvAddress, state
23:05:03 IP # [DBG] [ROUTER.LAN] onUpdate: activeV4AddrCount
23:05:03 IP # [INF] [eth0] Kernel >> Address departed: DHCP (172.16.60.4/19)
23:05:03 IP # [INF] [ROUTER.LAN.DHCP] State: revoking (release) | Event: down
23:05:03 IP # [INF] [ROUTER.LAN.DHCP] State: revoking (release) | Clear address
23:05:03 IP # [INF] [ROUTER.LAN.DHCP] State: revoking (release) | Action: restart
(release)
23:05:03 IP # [INF] [ROUTER.LAN.DHCP] State: revoking (release) | Action: release
dynamic DHCP address on dev eth0 forced
23:05:03 IP # [DBG] [eth0] Releasing DHCP address
23:05:03 IP # [INF] [ROUTER.LAN.DHCP] Restart Mode: release -> normal
23:05:03 IP # [INF] [ROUTER.LAN.DHCP] State: revoking (normal) -> released
23:05:03 IP # [DBG] [ROUTER.LAN.DHCP] onUpdate: address, restartMode, state
host(if-ip)[ROUTER.LAN]#ipaddress DHCP renew
23:05:15 IP # [INF] [ROUTER.LAN.DHCP] State: released (normal) | Event: renew
23:05:15 IP # [INF] [ROUTER.LAN.DHCP] State: released (normal) | Action: request
dynamic DHCP address on dev eth0
23:05:15 IP # [INF] [eth0] Starting DHCP client
23:05:15 IP # [INF] [eth0] DHCP Client << udhcpd -f -R -i eth0 -s /usr/bin/
DHCPInsert -C host -H host
23:05:15 IP # [DBG] [ROUTER.LAN.DHCP] onUpdate
23:05:15 IP # [INF] [eth0] DHCP Client >> udhcpd (v1.19.3) started
23:05:15 IP # [INF] [eth0] DHCP Client >> deconfig: interface="eth0"
23:05:15 IP # [INF] [eth0] DHCP Client: Release default route
23:05:15 IP # [INF] [eth0] DHCP Client >> Sending discover...
23:05:15 IP # [INF] [eth0] DHCP Client >> Sending select for 172.16.60.4...
23:05:15 IP # [INF] Lease of 172.16.60.4 obtained, lease time 86400
23:05:15 IP # [INF] [eth0] DHCP Client: Release IP address
23:05:15 IP # [INF] [ROUTER.LAN.DHCP] State: released (normal) | Event: dynamic
DHCP address released, reason=DHCP RELEASED
23:05:15 IP # [DBG] [ROUTER.LAN.DHCP] onUpdate
23:05:15 IP # [INF] [eth0] DHCP Client >> bound: interface="eth0",

```

```
ip="172.16.60.4", mask="19", router="172.16.32.1", dns="172.16.20.10 172.16.20.20"
23:05:15 IP # [INF] [eth0] DHCP Client: Apply IP address 172.16.60.4/19
23:05:15 IP # [INF] [eth1.1] DHCP Client >> Sending discover...
23:05:15 IP # [INF] [ROUTER.LAN.DHCP] State: released (normal) | Event: dynamic
DHCP address received: address=172.16.60.4/19, broadcast=
23:05:15 IP # [INF] [ROUTER.LAN.DHCP] State: released (normal) | Action: apply
address 172.16.60.4/19 to dev eth0
23:05:15 IP # [INF] [eth0] Kernel << ip addr add 172.16.60.4/19 broadcast + label
eth0:DHCP dev eth0
23:05:15 IP # [INF] [ROUTER.LAN.DHCP] State: released (normal) -> applying
23:05:15 IP # [DBG] [ROUTER.LAN.DHCP] onUpdate: address, recvAddress, state
23:05:15 IP # [INF] [eth0] DHCP Client: Apply default route via gateway
172.16.32.1
23:05:16 IP # [INF] [eth0] DHCP Client: Apply DNS server 172.16.20.10
23:05:16 IP # [INF] [eth0] DHCP Client: Apply DNS server 172.16.20.20
23:05:16 IP # [INF] [eth0] Kernel >> Address arrived: DHCP (172.16.60.4/19)
23:05:16 IP # [INF] [ROUTER.LAN.DHCP] State: applying (normal) | Event: up
23:05:16 IP # [INF] [ROUTER.LAN.DHCP] State: applying (normal) -> up
23:05:16 IP # [INF] [ROUTER.LAN.DHCP] State: applying (normal) | Action: incre-
ment active address counter
23:05:16 IP # [DBG] [ROUTER.LAN.DHCP] onUpdate: state
23:05:16 IP # [DBG] [ROUTER.LAN] onUpdate: activeV4AddrCount
```

Chapter 32 **DNS Configuration**

Chapter contents

Introduction	387
DNS Configuration Task List	387
Enabling the DNS Resolver	387
Enabling the DNS Relay	388

Introduction

The domain name system (DNS) enables users to contact a remote host by using easily remembered text labels (www.patton.com, for example) instead of having to use the host's numeric address (209.45.110.15, for example). When DNS names are entered as part of configuration commands or CLI exec mode commands in applications like Ping, Traceroute, or Tftp, the Patton device uses a DNS resolver component to convert the DNS names into the numeric address.

The Patton device can be configured as a caching DNS relay server to speed data transfers, acting as the DNS server for a private network. In this configuration, hosts in the network send their DNS queries to the Patton device, which checks to see if the DNS name is in its DNS resolver cache. If it finds the name in cache, the device uses the cached data to resolve the DNS name into a numeric IP address. If the name is not in cache, the query is forwarded on to a DNS server. When the device receives the answer from the server, it adds the name to the cache, and forwards it on to the host that originated the query. This process enables the Patton device to provide answers more quickly to often-queried DNS names, reducing the number of DNS queries that must be sent across the access link.

DNS Configuration Task List

The following sections describe how to configure the DNS component:

- Enabling the DNS resolver
- Enabling the DNS relay

Enabling the DNS Resolver

To enable the Patton device DNS resolver for manually configured DNS upstream servers, you must configure it with the address of one or more DNS servers that will be used to resolve DNS name queries. If multiple DNS servers are configured, the device will query each server in turn until a response is received.

There are two configuration modes: “dns-server” and “dns-client”.

- The “dns-server” mode is meant to configure the own DNS server of the Patton device, defining its own port, domain and host -> address static translations. For specific domains (which can be defined too), it can also work as a relay, forwarding the DNS queries to the configured DNS servers. This mode can be shut down to disable the DNS server.
- The “dns-client” mode is meant to define the DNS servers the Patton device will query when the “dns-server” mode cannot provide the required DNS resolution. This can happen when the “dns-server” mode is shut down, it does not have the required host-address translation set up, the domain is not listed in the ones configured in the DNS-relay, or those servers are not accessible. The “dns-client” mode cannot be shut-down.

For DNS servers being received through DHCP or PPP, no configuration is needed. These DNS servers will be stored regardless of the dns-server config, and will be used for DNS queries originating on the Patton device.

Manual entries for DNS servers are configured as follows:

Mode: dns-server

Step	Command	Purpose
1	node(dns-srv)#relay name-server domain <name> <IPv4 IPv6> [<port>]	Configures a DNS server used by the relay for the specific domain name.
Repeat step 1 for each additional DNS server you want to add		

Example: Configuring DNS servers

The following example shows how to add DNS servers to the Patton device DNS resolver and increase the size of the DNS cache to 100 entries.

```
node>enable
node#configure
node#dns-server name-server address 62.2.32.5
node#dns-server no shutdown
```

Mode: dns-client

Step	Command	Purpose
1	node(dns-client)#name-server <IPv4 IPv6>	Defines a DNS server used by the Patton device with the specific IP address.

Enabling the DNS Relay

To use DNS relay on the device, the DNS server has to be enabled (“no shutdown”).

The DNS resolver automatically learns domain name servers if it receives them through PPP or DHCP protocols, regardless if dns-server is set to shutdown or not.

You can verify that the DNS resolver has received domain name servers as well as the manually configured upstream DNS servers, by using the **show dns-client** command as follows:

```
node(cfg)#show dns
DNS Configuration
=====
Shutdown           : false
Name               :
Port               : 53

Name Server Table
=====
IP                 Domain          Port
192.168.1.1       192.168.1.1     53

DNS Host Information
=====
Host IP            Host Name
```

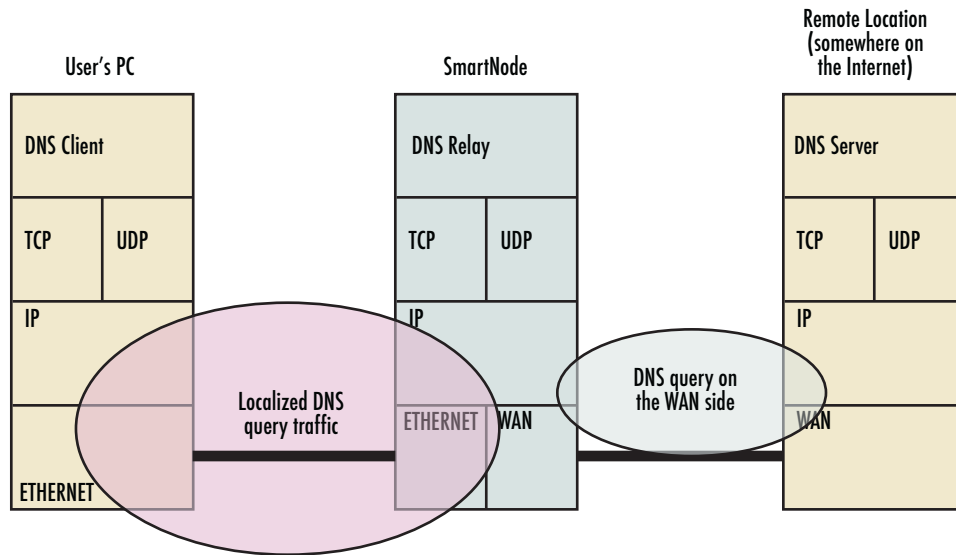



Figure 52. DNS relay diagram

Chapter 33 **SNTP Client Configuration**

Chapter contents

Introduction	391
NTP Client Configuration Task List	391
Enabling/Disabling the NTP Management Component	391
Enabling NTP Options	391
Examples	392
Run the following to see the NTP configuration settings	392
Run the following to see the NTP status	392

Introduction

This chapter describes how to configure Network Time Protocol (NTP) client. The NTP Management component enables users to setup an NTP time updating, given they are connected to the Internet. The NTP allows you to set several NTP Servers to allow for precise time keeping on the device. The NTP also allows you to listen to the NTP Broadcasts.

NTP Client Configuration Task List

The following sections describe how to configure the NTP component:

- Enabling/Disabling NTP Management component
- Configuring NTP options

Enabling/Disabling the NTP Management Component

Mode: Configure

Step	Command	Purpose
1	device(cfg)# ntp no shutdown	Enables the management component
2	device(cfg)# ntp shutdown	Disables the management component

Enabling NTP Options

Mode: Configure

Step	Command	Purpose
1	device(cfg)# ntp broadcastclient	Enables the broadcast client. NTP component supports NTP Broadcasts. The inverted command 'no' will disable the NTP server
2	device(cfg)# ntp <server>	Adds NTP Server to the DB.
3	device(cfg)# ntp <server> multicast	Optional multicast flag that will enable multicast NTP capabilities

Mode: Configure

Step	Command	Purpose
1	device(cfg)# show ntp config	Shows NTP Configuration
2	device(cfg)# show ntp status	Shows NTP Status

Examples

Run the following to see the NTP configuration settings

```
device(cfg)# show ntp config

NTP Status and Information
=====
Broadcast Client      : false
Shutdown              : false

NTP Servers
=====
192.168.0.2           : Unicast
96.47.67.105          : Unicast
```

Run the following to see the NTP status

The Active server will be shown with an arrow. The Server name is what the NTP config entry has resolved to. It may be different from what you've configured. That is how the protocol is meant to be operating:

```
device(cfg)# show ntp status

Active      Server                Delay      Offset     Jitter
=====
---->      hit-nxdomain.op             0.000     0.000     0.000
           nist1-nj.ustimi      23.244     0.575     1.592
```

Chapter 34 **SNMP Configuration**

Chapter contents

Introduction	394
Simple Network Management Protocol (SNMP)	394
SNMP Basic Components	394
SNMP Basic Commands	394
SNMP Management Information Base (MIB)	395
Network Management Framework	395
Identification of a Patton device via SNMP	396
SNMP Tools	396
SNMP Configuration Task List.....	396
Setting Basic System Information	396
Setting Access Community Information	399
Setting Allowed Host Information.....	400
Authentication and Encryption	400
Specifying the Default SNMP Trap Target.....	401
Displaying SNMP Related Information.....	401
Using the ManageEngine SNMP Utilities	402
Using the MibBrowser	402
Using the TrapViewer	403
Standard SNMP Version 1 Traps	406
SNMP Interface Traps	407

Introduction

This chapter provides overview information about Simple Network Management Protocol (SNMP) and describes the tasks used to configure those of its features supported.

Note SNMP v1/v2/v3 are supported in Trinity from version 3.7 onwards.

This chapter includes the following sections:

- Simple Network Management Protocol (SNMP)
- SNMP tools (see page 396)
- SNMP configuration task list (see page 396)
- Using the ManageEngine SNMP utilities (see page 402)
- Standard SNMP version 1 traps (see page 406)

Simple Network Management Protocol (SNMP)

The Simple Network Management Protocol (SNMP) is an application-layer protocol that facilitates the exchange of management information between network devices. It is part of the Transmission Control Protocol/Internet Protocol (TCP/IP) suite. SNMP enables network administrators to manage network performance, find and solve network problems, and plan for network growth.

SNMP Basic Components

An SNMP managed network consists of three key components: managed devices, agents, and network-management systems (NMSs).

A managed device is a network SN that contains an SNMP agent and resides on a managed network. Managed devices collect and store management information and make this information available to NMSs using SNMP. Managed devices, sometimes called network elements, can be routers and access servers, switches and bridges, hubs, computer hosts, or printers.

An agent is a network-management software module that resides in a managed device. An agent has local knowledge of management information and translates that information into a form compatible with SNMP.

An NMS executes applications that monitor and control managed devices. NMSs provide the bulk of the processing and memory resources required for network management. One or more NMSs must exist on any managed network.

SNMP Basic Commands

Managed devices are monitored and controlled using four basic SNMP commands: read, write, trap, and traversal operations.

- The read command is used by an NMS to monitor managed devices. The NMS examines different variables that are maintained by managed devices.
- The write command is used by an NMS to control managed devices. The NMS changes the values of variables stored within managed devices.
- The trap command is used by managed devices to asynchronously report events to the NMS. When certain types of events occur, a managed device sends a trap to the NMS.

- Traversal operations are used by the NMS to determine which variables a managed device supports and to sequentially gather information in variable tables, such as a routing table.

SNMP Management Information Base (MIB)

A Management Information Base (MIB) is a collection of information that is organized hierarchically. MIBs are accessed using a network-management protocol such as SNMP. They are comprised of managed objects and are identified by object identifiers.

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using the subset of abstract syntax notation one (ASN.1) defined in the SMI. In particular, an *object identifier*, an administratively assigned name, names each object type. The object type together with an object instance serves to uniquely identify a specific instantiation of the object. For human convenience, a textual string, termed the descriptor, to refer to the object type, is often used.

An object identifier (OID) world-wide identifies a managed object in the MIB hierarchy. The MIB hierarchy can be depicted as a tree with a nameless root, the levels of which are assigned by different organizations.

Network Management Framework

This section provides a brief overview of the current SNMP management framework. An overall architecture is described in RFC 2571 “An Architecture for Describing SNMP Management Frameworks.” The SNMP management framework has several components:

- Mechanisms for describing and naming objects and events for the purpose of management. The first version, Structure of Management Information (SMIv1) is described in RFC 1155 “Structure and Identification of Management Information for TCP/IP-based Internets”, RFC 1212 “Concise MIB Definitions”, RFC 1213 “Management Information Base for Network Management of TCP/IP-based Internets: MIB-II”, and RFC 1215 “A Convention for Defining Traps for use with the SNMP”. The second version, SMIv2, is described in RFC 2233 “The Interfaces Group MIB using SMIv2”, RFC 2578 “Structure of Management Information Version 2 (SMIv2)”, RFC 2579 “Textual Conventions for SMIv2”, and RFC 2580 “Conformance Statements for SMIv2”.
- Message protocols for transferring management information. The first version, SNMPv1, is described in RFC 1157 “A Simple Network Management Protocol (SNMP).” The second version, SNMPv2, which is not an Internet standards track protocol, is described in RFC 1901 “Introduction to Community-Based SNMPv2” and RFC 1906 “Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)”.
- Protocol operations for accessing management information. The first set of protocol operations and associated protocol data unit (PDU) formats is described in RFC 1157. The second set of protocol operations and associated PDU formats is described in RFC 1905 “Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)”.
- A set of fundamental applications described in RFC 2573 “SNMP Applications” and the view-based access control mechanism described in RFC 2575 “View-Based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)”.

Identification of a Patton device via SNMP

All product models have assigned sysObjectID.

Refer to the getting started guide of your product, or see the MIB definition file (.my) for sysObjectIDs.

SNMP Tools

Patton recommends the [ManageEngine](#).

Refer to section “Using the ManageEngine SNMP Utilities” on page 402 for more detailed information on how to use these tools.

SNMP Configuration Task List

To configure SNMP, perform the tasks described in the following sections. The tasks in the first three sections are required; the tasks in the remaining sections are optional, but might be required for your application.

- Setting basic system information (required) (see page 396)
- Setting access community information (required) (see page 399)
- Setting allowed host information (required) (see page 400)
- Specifying the default SNMP trap target (optional) (see page 401)
- Displaying SNMP related information (optional) (see page 401)

Setting Basic System Information

The implementation of the MIB-II system group is mandatory for all systems. By default, an SNMP agent is configured to have a value for any of these variables and responds to get commands from a NMS.

The following MIB II panels should be set:

- sysContact
- sysLocation
- sysName

The system sysContact object is used to define the contact person, together with information on how to contact that person.

Assigning explanatory location information to describe the system physical location (e.g. server room, wiring closet, 3rd floor, etc.) is very supportive. Such an entry corresponds to the MIB II system sysLocation object.

The name used for sysName should follow the rules for ARPANET host names. Names must start with a letter, end with a letter or digit, and have as interior characters only letters, digits, and hyphens. Names must be 63 characters or fewer. For more information, refer to RFC 1035.

This procedure describes how to set these MIB-II system group objects.

Mode: Administrator execution

Step	Command	Purpose
1	<code>device(cfg)#system contact name</code>	Sets the contact persons name

Step	Command	Purpose
2	device(cfg)#system location <i>location</i>	Sets the system location
3	device(cfg)#system hostname <i>hostname</i>	Sets the system hostname and command line prompt

If any of the command options *name*, *location*, or *hostname* has to be formed out of more than one word, the information is put in “double quotes”.

Note Enter an empty string “” to get rid of any of the system settings.

The MIB-II system group values are accessible for reading and writing via the following SNMP objects:

- .iso.org.dod.internet.mgmt.mib-2.system.sysContact
- .iso.org.dod.internet.mgmt.mib-2.system.sysName
- .iso.org.dod.internet.mgmt.mib-2.system.sysLocation

After setting these values according to 1 through 3 any SNMP MIB browser application should read the values using a get or get-next command as shown in [figure 53](#).

The procedure to use the SNMP MIB browser is:

- Enter the community string public into the Community field in the upper right corner of the window. For safety reasons each entered character is displayed with a “*”.
- Access any of the supported MIB system group object by using the GetNext button from the button bar of the window.

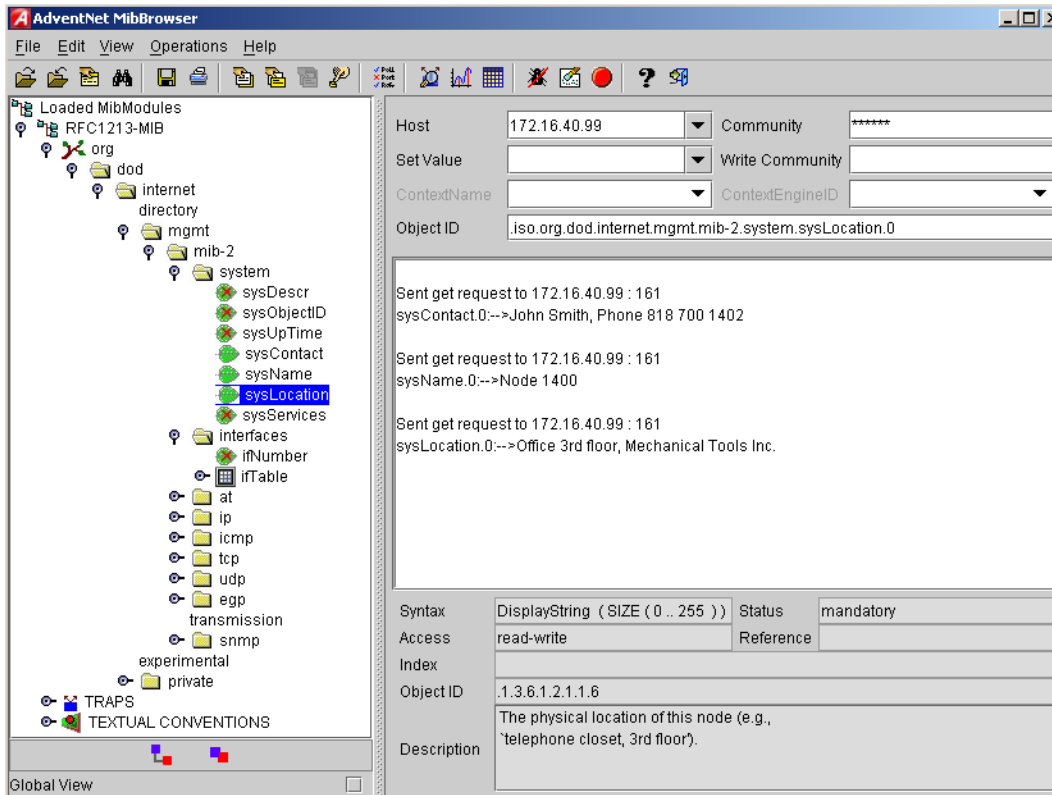


Figure 53. ManageEngine MibBrowser displaying some of the System Group objects

Example: Setting the system group objects

In the following example the system information is set for later access via SNMP. See [figure 53](#) for a typical MIB browser application accessing these MIB-II system group objects representing the system information.

```
device>enable
device#configure
device(cfg)#system contact "Bill Anybody, Phone 818 700 1504"
device(cfg)#system location "Wiring Closet, 3rd floor"
device(cfg)#system hostname "device"
(cfg)#
```

After entering a host name the prompt on the CLI no longer displays the IP address of the Ethernet port over which the Telnet session is running but shows the newly entered host name.

Setting Access Community Information

SNMP uses one or more labels called *community strings* to delimit groups of *objects* (variables) that can be viewed or modified on a device. The SNMP data in such a group is organized in a tree structure called a Management Information Base (MIB). A single device may have multiple MIBs connected together into one large structure, and various community strings may provide read-only or read-write access to different, possibly overlapping portions of the larger data structure. An example of a read-only variable might be a counter showing the total number of octets sent or received through an interface. An example of a read-write variable might be the speed of an interface, or the hostname of a device.

Community strings also provide a weak form of access control in earlier versions of SNMP version 1 and 2. SNMP version 3 provides much improved access control using strong authentication and should be preferred over SNMP version 1 and 2 wherever it is supported. If a community string is defined, then it must be provided in any basic SNMP query if the requested operation is to be permitted by the device. Community strings usually allow read-only or read-write access to the entire device. In some cases, a given community string will be limited to one group of read-only or read-write objects described in an individual MIB.

In the absence of additional configuration options to constrain access, knowledge of the single community string for the device is all that is required to gain access to all objects, both read-only and read-write, and to modify any read-write objects.

Note Security problems can be caused by unauthorized individuals possessing knowledge of read-only community strings so they gain read access to confidential information stored on an affected device. Worse can happen if they gain access to read-write community strings that allow unauthorized remote configuration of affected devices, possibly without the system administrators being aware that changes are being made, resulting in a failure of integrity and a possible failure of device availability. To prevent these situations, define community strings that only allow read-only access to the MIB objects should be the default.

Choosing community names is like choosing a password. Do not use easily guessed ones; do not use commonly known words, mix letters and other characters, and so on. If you do not intend to allow anyone to use SNMP write commands on your system, then you probably only need one community name.

This procedure describes how to define your own SNMP community.

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg)#snmp-server</code>	Enters snmp-server mode

Use the `no` command option to remove a SNMP community setting.

Example: Setting access community information

In the following example the SNMP communities for the default community `public` with read-only access and the undisclosed community `Not4evEryOne` with read/write access are defined. Only these valid communities have access to the information from the SNMP agent.

```
device(cfg)#snmp community public ro
```

```
device(cfg)#snmp community Not4evEryOne rw
```

Note If no community is set on your Patton device accessing any of the MIB objects is not possible!

Setting Allowed Host Information

If a host has to access SNMP MIB objects on a certain device, it explicitly needs the right to access the SNMP agent. Therefore a host needs an entry, which allows accessing the device. The host is identified by its IP address and has to use a certain community string for security precautions.

Note The community which is to be used as security name to access the MIB objects has to be defined prior to the definition of allowed hosts.

This procedure describes adding a host that is allowed to access the MIB of this system.

Mode: Configure

Step	Command	Purpose
1	device(cfg)#snmp host <i>IP-address-of-device</i> security-name <i>community</i>	Configures a host that with IP address <i>IP-address-of-device</i> can access the MIB, using the security name <i>community</i> .

Use the no command option to remove a SNMP allowed host setting.

Example: Setting allowed host information

In the following example the host with IP address *172.16.224.45* shall be able to access the MIB using community *public* as security name.

```
device(cfg)#snmp host 172.16.224.45 security-name public
```

This procedure describes adding an IP network that is allowed to access the MIB of this system.

Mode: Configure

Step	Command	Purpose
2	[node](snmp)#network <i><network-ip/netmask></i> security-name <i><community></i>	Configures an IP network that is allowed to access the MIB with the community rights as specified.

Authentication and Encryption

The authentication and encryption protocols can be configured globally for all users. The users have the option of SHA or MD5 for authentication and AES or DES for packet encryption. (If either are disabled, SNMPv3 will not work.) (NOTE: SNMPv1 and SNMPv2 cannot be explicitly disabled, however the service cannot be accessed if no HOSTS are defined.)

By default, all users in the local AAA database can assess SNMP; however this can be limited when defining the users.

Mode: snmp-server

Step	Command	Purpose
1	[no] authentication-protocol {sha md5}	Authentication protocol used to access service.
2	[no] encryption-protocol {aes des}	Encryption protocol used to access service.

Mode: configure

Step	Command	Purpose
1	[no] {superuser administrator operator} <user-name> {no-password password <password> [encrypted]} [terminal-type {console http snmp ssh telnet}]	Optionally select what type of service a user can access from.

Specifying the Default SNMP Trap Target

An SNMP trap is a message that the SNMP agent sends to a network management station. For example, an SNMP agent would send a trap when an interface's status has changed from up to down. The SNMP agent must know the address of the network management station so that it knows where to send traps. It is possible to define more than one SNMP trap target.

The SNMP message header contains a *community* field. The SNMP agent uses a defined community name, which is inserted in the trap messages header sent to the target. In most cases the target is a NMS, which only accepts a SNMP message header of a certain community.

This procedure describes how to define a SNMP trap target and enter community name.

Mode: Configure

Step	Command	Purpose
1	device(cfg)#snmp target IP-address-of-device security-name community	Configures a SNMP trap target with IP-address-of-hostname <i>device</i> that receives trap messages using the security name <i>community</i> on the target.

Use the no command option to remove s SNMP trap target setting.

Example: Specifying the default SNMP trap target

In the following example the NMS running on host with IP address 172.16.224.44 shall be defined as SNMP trap target. Since the NMS requires that SNMP message headers have a community of *Not4evEryOne* the security-name argument is set accordingly.

```
device(cfg)#snmp target 172.16.224.44 security-name Not4evEryOne
```

Displaying SNMP Related Information

Displaying the SNMP related configuration settings is often necessary to check configuration modifications or when determining the behavior of the SNMP agent.

This procedure describes how to display information and configuration settings for SNMP.

Mode: Configure

Step	Command	Purpose
1	device(cfg)#show snmp	Displays information and configuration settings for SNMP

Example: Displaying SNMP related information

This example shows how to display SNMP configuration information.

```
device(cfg)#show snmp

Hosts:
 172.16.224.44 security-name public

Targets:
 172.16.224.44 security-name Not4evEryOne

Communities:
 public access-right ro
 Not4evEryOne access-right rw
```

Using the ManageEngine SNMP Utilities

The ManageEngine SNMP utilities are a set of cross-platform applications and applets for SNMP and Web-based network management. These utilities can be used for device, element, application and system management. The following tools are the most useful:

- MibBrowser—used to view and operate on data available through a SNMP agent on a managed device
- TrapViewer—used to parse and view the received traps

The ManageEngine is a complete SNMP MibBrowser that enables the loading of MIBs, MIB browsing, walking a MIB tree, searching MIBs and performing all other SNMP-related functions to users.

Viewing and operating the data available through an SNMP agent on a managed device, e.g. a router, switch, hub etc., is made possible by using the MibBrowser.

The TrapViewer is a graphical tool to view the Traps received from one or more SNMP agents. The Trap viewer can listen to one or more port at a time and the traps can be sent from any host. Moreover the TrapViewer contains a Trap parser editor, which is a tool to create a trap parser file. The Trap viewer parses the file created using Trap parser editor to match each incoming traps with certain criteria. Since Traps typically contain cryptic information, which is not easily understandable to the users, trap parsers are required to translate or parse traps into understandable information.

Using the MibBrowser

Figure 54 on page 403 depicts the primary window of the ManageEngine MibBrowser. It consists of a menu bar, a toolbar, a left frame and a right frame.

The operations that can be performed by the MibBrowser are available in a series of buttons in the toolbar on top of the MibBrowser's main window. The toolbar can be hidden or made visible using the options available.

The menu bar has various options that perform the same operations as the options available in the toolbar.

The left frame holds the MIB tree. A MIB tree is a structure through which all the MIBs loaded can be viewed. The MIB tree component enables us to traverse through the tree, view the loaded MIBs and learn the definition for each SN. The ManageEngine MibBrowser allows loading additional MIB files in the text format (the “my” file contains enterprise specific MIB definitions).

The right frame has labeled text fields to specify the basic parameters like host, community etc. and a Result text area display to view the results.

There are three ways in which the primary window of the MibBrowser can be viewed. It can be viewed with the result display, MIB description panel or multi-variable bind panel in the right frame. The view can be altered in three ways.

- The desired view can be set by the options provided in the display menu item under the view menu. (View I Display I).
- The other way of altering the view is through the general settings panel in the settings menu item in the edit menu. (Edit I Settings)
- The same can be done through clicking the MibBrowser settings button on the toolbar. See [figure 54](#).



Figure 54. ManageEngine MibBrowser Settings Button on the Toolbar

By default the MIB description display and the result display are visible in the MibBrowser.

Using the TrapViewer

TrapViewer is a graphical tool to view the traps received from one or more SNMP agents. The TrapViewer can listen to one or more port at a time and the traps can be sent from any host.

Invoke the TrapViewer through the usage of the MibBrowser. To get to know more about the MibBrowser refer to section “Using the MibBrowser” on page 402. [Figure 55](#) is a screen shot of the TrapViewer.

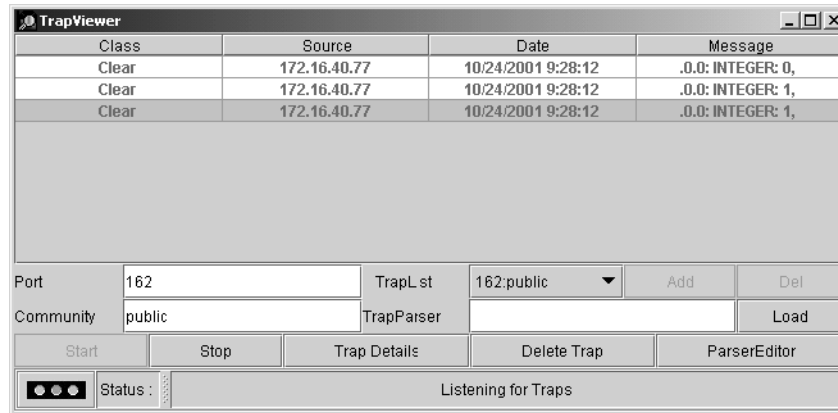


Figure 55. ManageEngine TrapViewer displaying received traps

The TrapViewer has a table that displays the trap information, the common parameters text fields where necessary information has to be entered and other options such as Start, Stop, Trap Details, Delete Trap and ParserEditor.

Follow these steps to work on the Trap Viewer and to know more about the available options:

- By default the value in the *Port* text field is 162. Enter the desired port in the field on which the viewer will listen.
- The default value in the *Community* text field is public. Set the community of the incoming traps as desired, depending on the SNMP configuration.
- Click on *Add* button to add the port and community list on which the trap has to listen to. This is visible in the *TrapList* combo box.
- The port and community list can be deleted by clicking on the *Del* button.
- When you need to load a trap parser file, click on the *Load* button, which will open up a dialog box, from which you can load the parser file.
- In order to receive the traps now, click on the *Start* button. Upon clicking this button, TrapViewer begins to receive traps according to the as-specified port and community.
- Once received, the traps are listed in the trap table of the TrapViewer. By default, the trap table has the following four columns:
 - *Class* that defines the severity of the trap.
 - *Source* that displays the IP address of the source from where the traps were sent.
 - *Date* that shows the date and time when the trap was received.
 - *Message* that by default has the object identifier format (sequence of numeric or textual labels on the SNs along a path from the root to the object) of the trap if any, or it is blank.
- The details of the traps can be viewed by clicking the *Trap Details* button or right click the trap in the trap table and select the option *View Trap Details*. [figure 56](#) show the screen of such a trap details window.

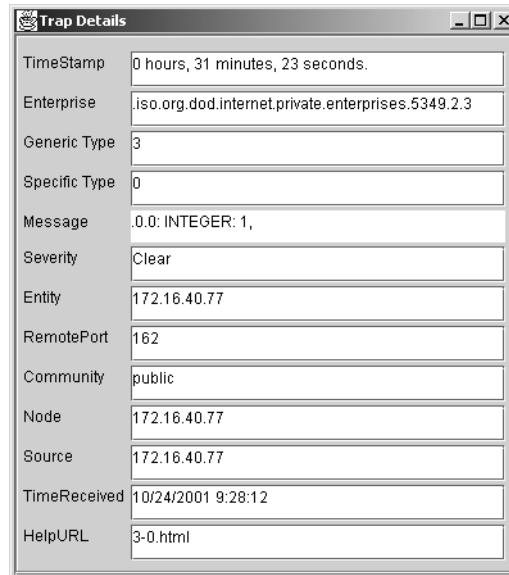


Figure 56. ManageEngine Trap Details window of TrapViewer

The various details available in the Trap Details window are listed in [table 26](#):

Table 26. Details available in the Trap Details window

Trap Details	Description
TimeStamp	The TimeStamp is a 32-bit unsigned value indicating the number of hundredths-of-a-second that have elapsed since the (re)start of the SNMP agent and the sending of the trap. This field shows the value stored in the MIB-II sysUpTime variable converted into hours, minutes and seconds.
Enterprise	This field shows the OID of the management enterprise that defines the trap message. The value is represented as an OBJECT IDENTIFIER value and has a variable length.
Generic Type	The Generic type value is categorized and numbered 0 to 6. They are 0-coldStart, 1-warmStart, 2-linkDown, 3-linkUp, 4-authenticationFailure, 5-egpNeighborLoss. The trap type value 6 is identified as enterprise-specific value. This field shows the value based on the type of trap.
Specific Type	The specific trap type indicates the specific trap as defined in an enterprise-specific MIB. If the Generic type value is 6, then this field shows a value greater than 0. If the generic type value is a value other than 6, then the field shows a value 0. This field can have values from 0 to 2147483647.
Message	This is a text field. By default, this field will always contain the Varbinds in the Trap PDU. This can be substituted with text.
Severity	This field shows the Severity or the intensity of the trap. They could be 0-All, 1-Critical, 2-Major, 3-Minor, 4-warning, 5-Clear and 6-info.
Entity	The source IP address from which the Trap was sent is displayed here.
RemotePort	This field reveals the port on which the Trap was sent by the originator.
Community	The Community string is displayed here.
device	Source

Table 26. Details available in the Trap Details window (Continued)

Trap Details	Description
TimeReceived	This displays the Date and Time when the trap was received.
HelpURL	The URL shown here gives more details of the received trap. By default, the URL file name is <generic-type value> - <specific-type value>.html

You can stop the listening by clicking the *Stop* button.

When you need to delete the trap, select the trap to be deleted and click the *Delete Trap* button or right click on the trap in the trap table and select option *Delete the Selected Rows*.

Yet another option in the Trap Viewer is the *ParserEditor*. The TrapViewer can filter incoming traps according to certain criteria called the parser criteria. The configuration of the criteria is made possible by using the parser editor. Refer to the ManageEngine SNMP Utilities documentation for a detailed description of the parser editor configuration and its use.

Standard SNMP Version 1 Traps

The following standard SNMP version 1 traps are supported. The descriptions are taken from RFC 1215 “Convention for defining traps for use with the SNMP”.

```
warmStart TRAP-TYPE
ENTERPRISE snmp
DESCRIPTION
"A warmStart trap signifies that the sending protocol entity is reinitializing
itself such that neither the agent configuration nor the protocol entity implemen-
tation is altered."
::= 1
```

```
linkDown TRAP-TYPE
ENTERPRISE snmp
VARIABLES { ifIndex }
DESCRIPTION
"A linkDown trap signifies that the sending protocol entity recognizes a failure in
one of the communication links represented in the agent's configuration."
::= 2
```

Note The linkDown trap is not sent if any of the ISDN ports has gone down.

```
linkUp TRAP-TYPE
ENTERPRISE snmp
VARIABLES { ifIndex }
DESCRIPTION
"A linkUp trap signifies that the sending protocol entity recognizes that one of
the communication links represented in the agent's configuration has come up."
::= 3
```

Note The linkUp trap is not sent if any of the ISDN ports has come up.

```

authenticationFailure TRAP-TYPE
ENTERPRISE snmp
DESCRIPTION
"An authenticationFailure trap signifies that the sending protocol entity is the
addressee of a protocol message that is not properly authenticated. While implemen-
tations of the SNMP must be capable of generating this trap, they must also be capa-
ble of suppressing the emission of such traps via an implementation-specific
mechanism."
 ::= 4

```

Note The authenticationFailure trap is sent after trying to access any MIB object with a SNMP community string, which does not correspond to the system setting.

```

coldStart TRAP-TYPE
ENTERPRISE snmp
DESCRIPTION
"A coldStart trap signifies that the sending protocol entity is reinitializing
itself such that the agent's configuration or the protocol entity implementation
may be altered."
 ::= 0

```

Note The standard SNMP version 1 trap coldStart as listed below is *not* supported. After powering up, a warmStart trap message is sent if any trap target host is defined.

SNMP Interface Traps

The Patton device sends Interface Traps (*linkUp*, *linkDown*) when the status of logical or physical interfaces change. Logical interfaces are interfaces defined in the IP context and CS context. Physical interfaces are ports.

The Patton device adds an entry to event log for each Interface Traps it sends:

```

device(cfg)#show log event

...
2002-09-06T14:54:35 : LOGINFO : Link up on interface sip_60.
2002-09-06T14:54:35 : LOGINFO : Link up on interface sip_30.
2002-09-06T14:54:35 : LOGINFO : Link up on interface isdn20.
2002-09-06T14:54:38 : LOGINFO : Link up on interface ETH00.
2002-09-06T14:54:38 : LOGINFO : Link up on interface ETH01.
2002-09-06T14:54:39 : LOGINFO : Link up on interface eth00.
2002-09-06T14:54:39 : LOGINFO : Link up on interface eth01.
2002-09-06T14:56:02 : LOGINFO : Link up on interface SLOT2:00 ISDN D
2002-09-10T14:21:20 : LOGINFO : Link down on interface SLOT2:00 ISDN
...

```

Chapter 35 **Public-Key Infrastructure (PKI)**

Chapter contents

Introduction	409
Overview	409
Architecture	409
Symmetric encryption and the key-distribution problem	410
Asymmetric encryption	411
CA-signed certificate enrollment	412
Self-signed certificate enrollment	413
PKI-related commands	414
Private-key handling	414
Public-key handling	415
Certificate-request handling	415
Own-certificate handling	418
Trusted-certificate handling	420
Generated default files	422

Introduction

This chapter provides an overview on how to set up the public-key infrastructure on a Patton device. PKI deals with the creation, management and deployment of keys and certificates, which is an intricate task. Therefore, this chapter first gives an introduction on general PKI concepts, before it discusses in detail how to configure a Patton device.

Overview

As business is moving forward, the expectation for secure communication over the public Internet is becoming more and more important. There are three primary security vulnerabilities of communications over a publicly accessible network:

- Eavesdropping: intruder captures the data transmission between two parties during communications.
- Identity theft: intruder gains illegal access by posing as an individual who actually can access secured resources.
- Man-in-the-Middle: intruder interrupts a dialogue and modifies the data between the two parties. The intruder could take over the entire session in the worst case.

Public Key Infrastructure (PKI) strongly reduces these risks. It provides a hierarchical framework for managing the digital security attributes of entities that will engage in secured communications.

Architecture

A PKI consists of the following elements (see [figure 57](#) on page 410):

- A Certificate Authority (CA) that both issues and verifies the digital certificates. The primary role of the CA is to digitally sign and publish the public key bound to a given user. This is done using the CA's own private key, so that trust in the user's key relies on one's trust in the validity of the CA's key.
- A Registration Authority (RA) which verifies the identity of users requesting information from the CA.
- Third-party Validation Authority (VA) which provides information on behalf of CA.

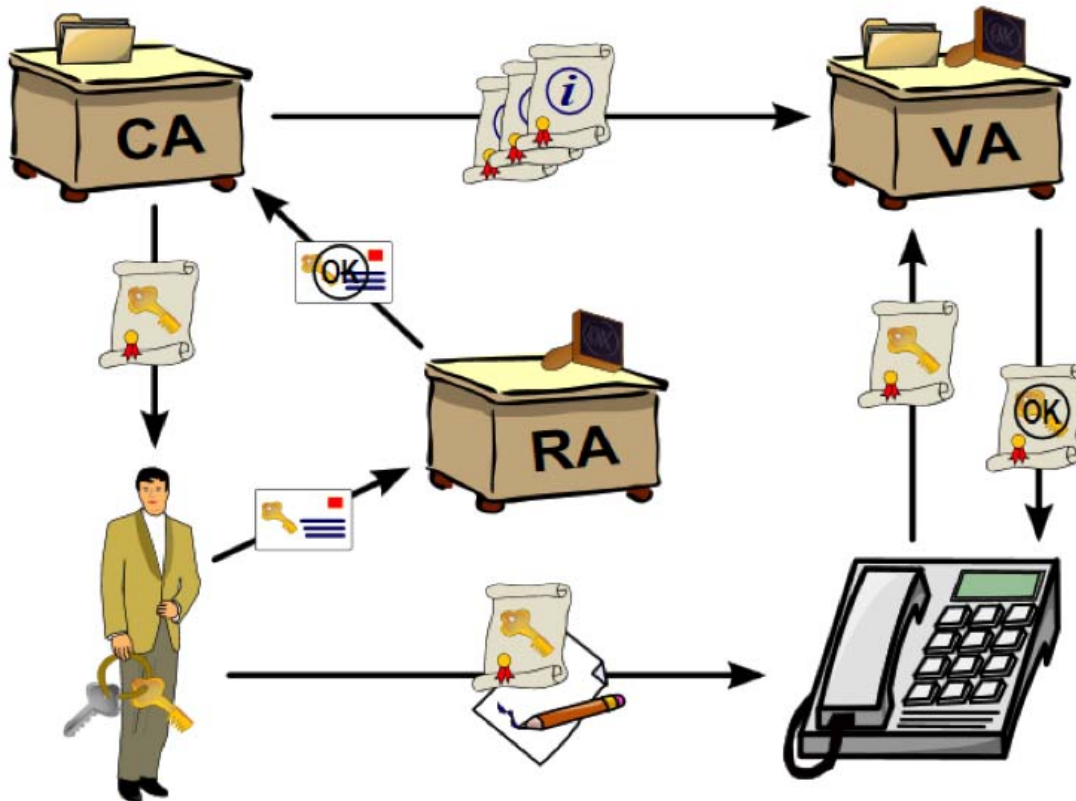


Figure 57. PKI Architecture

Symmetric encryption and the key-distribution problem

Symmetric encryption may also be referred to as shared key or shared secret encryption. In symmetric encryption, a single key is used both to encrypt and decrypt traffic (see [figure 58](#)). Symmetric encryption algorithms can be extremely fast, and their relatively low complexity allows for easy implementation in hardware. However, they require that all hosts participating in the encryption have already exchanged the secret key through some external means. Because of that one needs a secure channel to disclose these keys which will lead to asymmetric encryption.

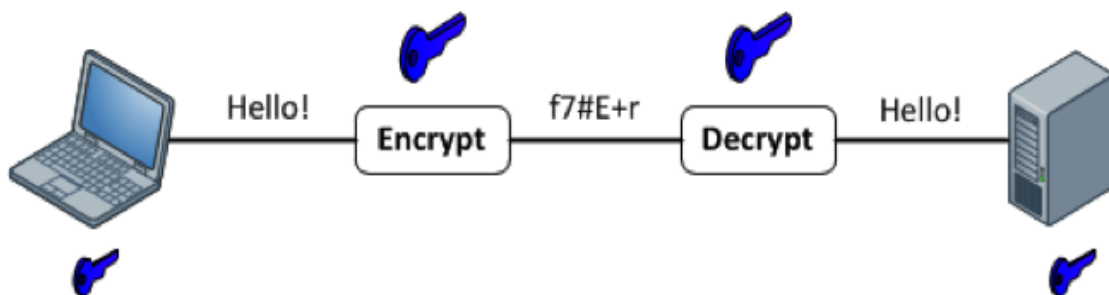


Figure 58. Symmetric Encryption

Asymmetric encryption

Compared to the symmetric way, asymmetric encryption imposes a high computational burden, and tends to be much slower. This property makes this way of communication less efficient if the amount of data is huge but definitely has one advantage: Its major strength is the ability to establish a secure channel over a non-secure medium (for example, the Internet). Let's assume that two participants have decided to talk to each other securely. As a first step each of them must generate its private/public key pair (see figure 59).

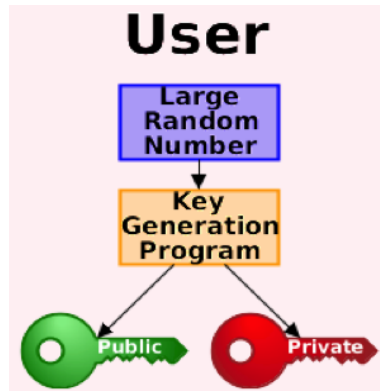


Figure 59. Private/Public Key Generation

The key generation program is fed by a large random number, which is different for each participant; therefore, the public and private keys will also differ.

Refer to figure 60 for an illustration of the usage of public/private key pair in asymmetric cryptography. The two parties will exchange their public keys (contained in the certificate), but will not disclose their private keys. The sending party will use the public key of the receiving party to encrypt message data and forward the encrypted data to the other party. The receiving party will then decrypt it with its private key. Data that is encrypted with the public key can be decrypted with the corresponding private key, and vice versa. However, data encrypted with the public key cannot be decrypted with the public key. This prevents someone from compromising the encrypted data after acquiring both public keys by sniffing on the certificate exchange.

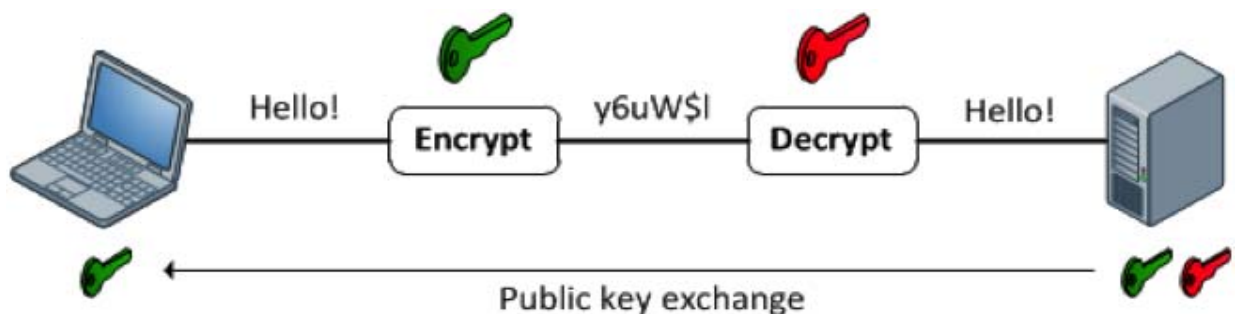


Figure 60. Asymmetric Encryption

If the secure channel is established it is possible to exchange a symmetric session key, which is used to actually encrypt and decrypt data. This allows for a more efficient communication.

CA-signed certificate enrollment

The mentioned public-key exchange happens via certificates. Enrollment is the process of obtaining these certificates.

The enrollment process looks like the following for an end host (see [figure 61](#)):

1. The end host generates a private/public key pair.
2. The end host generates a certificate request, which it forwards to the CA or RA.
3. Manual, human intervention is required to approve the enrollment request, which is received by CA or RA.
4. After the CA or RA operator approves the request, the CA or RA signs the certificate request with its own private key and returns the completed certificate to the end host.
5. The end host writes the certificate into a non-volatile storage area.



Figure 61. Certificate Enrollment Process

Example 1: Generate a private key and certificate request

This example illustrates how the above process is executed on a Patton device.

1. `node(cfg)#generate pki:private-key/key1 key-length 1024`
2. `node(cfg)#generate pki:certificate-request/request1 private-key pki:private-key/key1 country CH state Bern locality Bern organization Patton-Inalp organization-unit RND common-name 172.16.55.41`
3. `node(cfg)#export pki:certificate-request/request1`

Example 2: Generate all files on the CA server and import them

As an alternative, you may import the private key from a TFTP server and generate the certificate request offline.

1. `node(cfg)#copy tftp://server-ip-address/key1 pki:private-key/key1`
2. Generate the...
3. ...certificate request offline.

4. This certificate request has to be manually signed by the Certificate Authority.
5. `node(cfg)#copy tftp://server-ip-address/cert1 pki:certificate/cert1`

Self-signed certificate enrollment

The mentioned public key exchange happens via certificates. Enrollment is the process of obtaining these certificates.

In some cases it is enough to generate a specific self-signed certificate. In a web-of-trust certificate scheme there is no central CA, and so identity certificates for each user can be self-signed. As the name tells it is an identity certificate that is signed by the same entity whose identity it certifies. The generation process looks like the following for an end host (see [figure 62](#)):

1. The end host generates a private/public key pair.
2. The end host generates a certificate request.
3. The end host signs the certificate request with its own private key and writes the certificate into a non-volatile storage area.

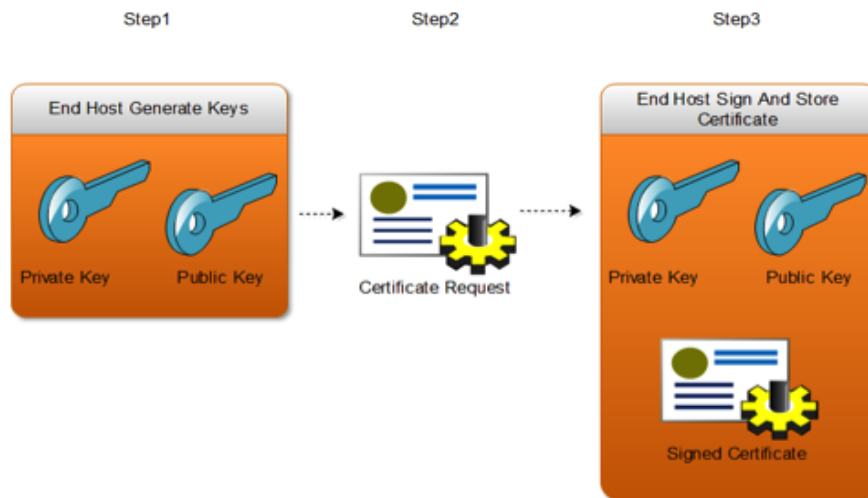


Figure 62. Self-signed Certificate Process

Example 1: Generate a private key and self-signed certificate

This example illustrates how the above process is executed on a Patton device.

1. `node(cfg)#generate pki:private-key/key1 key-length 1024`
2. `node(cfg)#generate pki:certificate-request/request1 private-key pki:private-key/key1 country CH state Bern locality Bern organization Patton-Inalp organization-unit RND common-name 172.16.55.41`
3. `node(cfg)#generate pki:certificate/cert1 private-key pki:private-key/key1 validity-period 3650`

Example 2: Import a private key and a self-signed certificate

As an alternative, you may import the private key and self-signed certificate from a TFTP server.

1. `node(cfg)#copy tftp://server-ip-address/key1 pki:private-key/key1`
2. Create and sign the certificate request offline
3. `node(cfg)#copy tftp://server-ip-address/cert1 pki:certificate/cert1`

PKI-related commands

This section introduces and discusses all PKI-related commands in Trinity:

- Private-key handling (see page 414)
- Public-key handling (see page 415)
- Certificate-request handling (see page 415)
- Own-certificate handling (see page 418)
- Trusted-certificate handling (see page 420)
- Generated default files (see page 422)

Private-key handling

Asymmetric encryption requires a private and a public key for encryption and decryption. They can only be generated and deleted together. With the **generate** command it is possible to generate an RSA key with a specified length. A longer modulus length corresponds to a higher level of security but requires more computational resources for key generation and connection handshaking. Keys are mainly used to generate or sign certificate requests.

In order to make the Private Key Infrastructure easier to use a couple of default files will be generated at the first startup. One of them is a private key named *DEFAULT* with a modulus length of 512 bits. The file is generated only once, it is immutable and can be used for basic scenarios (see chapter 56, “Secure SIP Applications” on page 725, for example).

Mode: Administrator exec

Step	Command	Purpose
1	<code>node(cfg)#generate pki:private-key/name [key-length <512 768 1024 2048>]</code>	Generates a private key. Note that it implicitly generates a public key as well.
2	<code>node(cfg)#copy tftp://node-ip-address/filename pki:private-key/name</code>	Imports a private key. Note that it implicitly imports a public key as well. A content check is applied: If the file is not a valid private key it will not be imported.
3	<code>node(cfg)#erase pki:private-key/name</code>	Erases a private key.

Example: Create a private key locally

```
node>enable
node#configure
node(cfg)#generate pki:private-key/key1 key-length 1024
```

```

Generating RSA private key, 1024 bit long modulus
.....+++++
.+++++
e is 65537 (0x10001)
writing RSA key

```

Example: Import a private key from a TFTP server

```

node>enable
node#configure
node(cfg)#copy tftp://172.16.55.1/key1 pki:private-key/key2

```

Example: Erase the two private keys

```

node>enable node#configure
node(cfg)#erase pki:private-key/key2
node(cfg)#erase pki:private-key/key1

```

Public-key handling

Asymmetric encryption requires a private and a public key for encryption and decryption. They can only be generated and deleted together with the **generate** command introduced above.

Mode: Administrator exec

Step	Command	Purpose
1	node(cfg)#export pki:public-key/name	Displays a public key in Base64 format, ready to be copied from the terminal for export.
2	node(cfg)#copy pki: public -key/name tftp://node-ip-address/filename	Exports a public key to a TFTP server

Example: Display a public key in Base64 format

```

node>enable node#configure
node(cfg)#export pki:public-key/key1
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCf7r3uLPuPbsVPX0mSOvV3Og+c
WZhf99M/OsZcstpZqxAlRq3rERiueC3SSDAq2lfEC+3Xgcs/uP/nfTL0IRPzvDpu
By7e2uHUnyNJD4HtirH1006+CAa0wTFdv1Ctv/Myaloq7tLulDx+KNVjI61YLKgw
jkfoc69A8bIs4Sh6+wIDAQAB
-----END PUBLIC KEY-----

```

Example: Export a public key to a TFTP server

```

node>enable
node#configure
node(cfg)#copy pki:public-key/key1 tftp://172.16.55.1/key1

```

Certificate-request handling

The key exchange happens via certificates. In order to create a certificate we have to generate a certificate request. All the parameters of the **generate** command are mandatory and have the following meaning:

- **private-key:** The request has to store a key which will be sent to the peer, which is the public key that corresponds to the specified private key. This public key will be the decryption key of the peer. If no private key is specified, the *DEFAULT* private key will be used.
- **country:** The certificate issuer's country of residence. If nothing is set then empty string will be used.
- **state:** The certificate owner's state of residence. If nothing is set then empty string will be used.
- **locality:** The certificate issuer's locality. If nothing is set then empty string will be used.
- **organization:** The organization to which the certificate issuer belongs. If nothing is set then empty string will be used.
- **organization-unit:** The name of the organizational unit to which the certificate issuer belongs. If nothing is set then empty string will be used.
- **common-name:** The certificate owner's common name. This has to be the IP address or the fully qualified DNS domain name ("im.example.org", "mail.example.net", and "www.example.com", respectively) of the local SIP gateway. The peer side can check whether these parameters match. If nothing was is then empty string will be used.

Mode: Administrator exec

Step	Command	Purpose
1	<code>node(cfg)#generate pki:certificate-request/name private-key <pki:private-key/name> country <country> state <state> locality <locality> organization <organization> organization-unit <organization-unit> common-name <common-name></code>	Generates a certificate request.
2	<code>node(cfg)#export pki: certificate-request/name</code>	Displays a certificate request in Base64 format, ready to be copied from the terminal for export.
3	<code>node(cfg)#show pki: certificate-request/name</code>	Shows a certificate request logical content.
4	<code>node(cfg)#copy pki: certificate-request/name tftp://node-ip-address/ filename</code>	Exports a certificate request to a TFTP server.
5	<code>node(cfg)#copy tftp://node-ip-address/ filename pki: certificate-request/name</code>	Imports a certificate request from a TFTP server. A content check will be applied: If the file is not a valid certificate request then it will not be imported.
6	<code>node(cfg)#erase pki: certificate-request/name</code>	Erases a certificate request.

Example: Generate a certificate request locally

```
node>enable node#configure node(cfg)#generate pki:certificate-request/request1 private-key
```

```
pki:private-key/key1 country CH state Bern locality Bern organization Inalp organi-
zation-unit RND common-name 172.16.55.41
```

Example: Display a certificate request in Base64 format

```
node>enable node#configure node(cfg)#export pki:certificate-request/request1
-----BEGIN CERTIFICATE REQUEST-----
MIIBoDCCAQkCAQAwYDELMAkGA1UEBhMCQ0gxDTALBgNVBAGMBEJlcm4xDTALBgNV
BACMBEJlcm4xDTALBgNVBAOMBuluYWxwMQwwCgYDVQQLDANStkQxFTATBgNVBAMM
DDE3Mi4xNi41NS40MTCBnzANBkgqhkiG9w0BAQEFAAOBjQAwgYkCgYEA+697iz7
j27FT19JkjrdzoPnFmYX/fTPzrGXLLaWasQJUat6xEYrngt0kgwKtpXxAvtl4HL
P7j/530y9CET87w6bgcu3trhlJ8jSQ+B7Yqx9TjuvvggGtMEExXb5Qrb/zMmtaKu7S
7pQ8fijVYyOtWCyhsI5H6HOvQPGyLOEoEvsCAwEAAaAAMA0GCSqSIB3DQEBBQUA
A4GBAGh3UJjIdJWyo+YpPmjIZffPBfav4JeqFNrHs2tWAep7lbsD2dYZ+pzFByVc
pu0U5Ioaptk7VmRvTRlDQpoYED/KntyXDv3SggB3Mf7cGq3xTZloqhXZNzN3PbJl
+kf16+4zB7Hlghfd//tdEhEUCJVMaWsYUWD85ur6yH8uolItv
-----END CERTIFICATE REQUEST-----
```

Example: Display the logical content of a certificate request

```
node>enable node#configure node(cfg)#show pki:certificate-request/request1
Certificate Request:
Data:
Version: 0 (0x0)
Subject: C=CH, ST=Bern, L=Bern, O=Inalp, OU=RND, CN=172.16.55.41
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
Public-Key: (1024 bit)
Modulus:
00:9f:ee:bd:ee:2c:fb:8f:6e:c5:4f:5f:49:92:3a:
f5:77:3a:0f:9c:59:98:5f:f7:d3:3f:3a:c6:5c:b2:
da:59:ab:10:25:46:ad:eb:11:18:ae:78:2d:d2:48:
30:2a:da:57:c4:0b:ed:d7:81:cb:3f:b8:ff:e7:7d:
32:f4:21:13:f3:bc:3a:6e:07:2e:de:da:e1:d4:9f:
23:49:0f:81:ed:8a:b1:f5:38:ee:be:08:06:b4:c1:
31:5d:be:50:ad:bf:f3:32:6b:5a:2a:ee:d2:ee:94:
3c:7e:28:d5:63:23:ad:58:2c:a1:b0:8e:47:e8:73:
af:40:f1:b2:2c:e1:28:7a:fb
Exponent: 65537 (0x10001)
Attributes:
a0:00
Signature Algorithm: sha1WithRSAEncryption
68:77:50:98:c8:74:95:b2:a3:e6:29:3e:68:c8:65:f7:cf:05:
f6:af:e0:97:aa:14:da:c7:b3:6b:56:01:ea:7b:95:bb:03:d9:
d6:19:fa:9c:c5:07:25:5c:a6:ed:14:e4:8a:1a:a6:d9:3b:56:
64:6f:4d:19:43:42:9a:18:10:3f:ca:9e:dc:97:0e:fd:d2:82:
06:f7:31:fe:dc:1a:ad:f1:4d:99:68:aa:15:d9:37:33:77:3d:
b2:65:fa:47:f5:eb:ee:33:07:b1:f5:81:f7:7f:fe:d7:44:84:
45:02:25:53:00:5a:c6:14:58:3f:39:ba:be:b2:1f:cb:a8:94:
8b:6f
```

Example: Export a certificate request to a TFTP server

```
node>enable
node#configure
```

```
node(cfg)#copy pki:certificate-request/request1 tftp://172.16.55.1/request1
```

Example: Import a certificate request from a TFTP server

```
node>enable
node#configure
node(cfg)#copy tftp://172.16.55.1/request1 pki:certificate-request/request2
```

Example: Erase the two certificate requests

```
node>enable
node#configure
node(cfg)#erase pki:certificate-request/request2
node(cfg)#erase pki:certificate-request/request1
```

Own-certificate handling

A certificate chain is transmitted to a remote peer when this peer wants to authenticate the local user/device. A certificate chain is formed from the following certificates:

- A personal certificate, which identifies the local user or device.
- Zero or more intermediate certificates, which are certificates that identify intermediary Certificate Authorities between the root certificate and the personal certificate.
- Zero or one root certificate, provided by a trusted third-party Certificate Authority (CA).

The ordered list of certificates, starting from the personal certificate and ending at the root CA, usually denotes a delegation of trust where the root CA trusts intermediate CA1, which in turns trusts intermediate CA2, which in turns trusts intermediate CA3 and so on, up to the intermediate CA that has issued the personal certificate to the local user or device, which trusts the user's/device's identity. With the following commands only single certificates can be handled; the chain of certificates must be formed later in the TLS profile.

Mode: Administrator exec

Step	Command	Purpose
1	node(cfg)#generate pki:own-certificate/name [private-key <pki:private-key/name>] [validity-period <days>]	Generates an own certificate.
2	node(cfg)#export pki:own-certificate/name	Displays an own certificate in Base64 format, ready to be copied from the terminal for export.
3	node(cfg)#show pki:own-certificate/name	Shows an own certificate logical content.
4	node(cfg)#copy pki:own-certificate/name tftp://node-ip-address/filename	Exports an own certificate to a TFTP server.
5	node(cfg)#copy tftp://node-ip-address/filename pki:own-certificate/name	Imports an own certificate from a TFTP server. A content check will be applied: If the file is not a valid certificate then it will not be imported.
6	node(cfg)#erase pki:own-certificate/name	Erases an own certificate.

Example: Generate a self-signed certificate locally

```
node>enable node#configure
node(cfg)#generate pki:own-certificate/cert1 pki:certificate-request/request1 private-key pki:private-key/key1 validity-period 365
Signature ok
subject=/C=CH/ST=Bern/L=Bern/O=Inalp/OU=RND/CN=172.16.55.41
Getting Private key
```

Example: Display a certificate in Base64 format

```
node>enable node#configure node(cfg)#export pki:own-certificate/cert1
-----BEGIN CERTIFICATE-----
MIICNzCCAAaACCQCWwsjV9Z+l7zANBgkqhkiG9w0BAQUFAADBGMQswCQYDVQQGEwJD
SDENMAsGA1UECAwEQmVybjeENMAsGA1UEBwwEQmVybjeEOMAwGA1UECgwFSW5hbHhax
DDAKBgNVBAsMA1JORDEVMBMGAlUEAwMMTCyLjE2LjU1LjQxMB4XDTEzMDcwMjA4
MjI1OjE0f0XDTE0MDcwMjA4MjI1OjE0f0YDELMAkGA1UEBhMCQ0gxDTALBgNVBAGM
BEJlcm4xDTALBgNVBACMBEJlcm4xDjAMBGNVBAoMBUluYWxwMQwwCgYDVQQLDANStkQx
FTATBgNVBAMMDDE3Mi4xNi41NS40MTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkC
gYEAan+697iz7j27FT19Jkjr1dzoPnFmYX/fTPzrGXLLaWasQUat6xEYrngt0kgw
KtpXxAvt14HLP7j/530y9CET87w6bgcu3trh1J8jSQ+B7Yqx9TjuvvgGtMExXb5Q
rb/zMmtaKu7S7pQ8fiJVYyOtWCyhsI5H6HOvQPGyLOeovsCAwEAATANBgkqhkiG
9w0BAQUFAAOBQAY2vYzZWxvkPspAzpCmxwC/YQkvfIhzAW8cbXpLX7I+pbQIYhr
N5BbyLGVTk7WWlw42jmsP+zZNwHqpJdCsd4kxOFmNWIBZSY2b0oiiX4rmgMCxohf
uHk0tzKfLHVzMVSPgiaCfeAZ6hfQU8MMe8YxJ8hPc/pQYu5aneKV4zAo3w==
-----END CERTIFICATE-----
```

Example: Display the logical content of a certificate

```
node>enable node#configure node(cfg)#show pki:own-certificate/cert1
Certificate:
Data:
Version: 1 (0x0)
Serial Number:
94:5a:c8:d5:f5:9f:a5:ef
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=CH, ST=Bern, L=Bern, O=Inalp, OU=RND, CN=172.16.55.41
Validity
Not Before: Jul 2 08:22:58 2013 GMT
Not After : Jul 2 08:22:58 2014 GMT
Subject: C=CH, ST=Bern, L=Bern, O=Inalp, OU=RND, CN=172.16.55.41
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
Public-Key: (1024 bit)
Modulus:
00:9f:ee:bd:ee:2c:fb:8f:6e:c5:4f:5f:49:92:3a:
f5:77:3a:0f:9c:59:98:5f:f7:d3:3f:3a:c6:5c:b2:
da:59:ab:10:25:46:ad:eb:11:18:ae:78:2d:d2:48:
30:2a:da:57:c4:0b:ed:d7:81:cb:3f:b8:ff:e7:7d:
32:f4:21:13:f3:bc:3a:6e:07:2e:de:da:e1:d4:9f:
23:49:0f:81:ed:8a:b1:f5:38:ee:be:08:06:b4:c1:
31:5d:be:50:ad:bf:f3:32:6b:5a:2a:ee:d2:ee:94:
3c:7e:28:d5:63:23:ad:58:2c:a1:b0:8e:47:e8:73:
af:40:f1:b2:2c:e1:28:7a:fb
Exponent: 65537 (0x10001)
Signature Algorithm: sha1WithRSAEncryption
```

```
18:da:f6:33:65:65:ef:90:fb:29:03:3a:42:9b:1c:02:fd:84:
24:bd:f2:21:cc:05:bc:71:b5:e9:2d:7e:c8:fa:96:d0:21:88:
6b:37:90:5b:c8:b1:95:4e:4e:d6:5a:5c:38:da:39:ac:3f:ec:
d9:37:01:ea:a4:97:42:b1:de:24:c4:e1:66:35:62:01:65:26:
36:6f:4a:
```

Example: Export a certificate to a TFTP server

```
node>enable
node#configure
node(cfg)#copy pki:own-certificate/cert1 tftp://172.16.55.1/cert1
```

Example: Import a certificate from a TFTP server

```
node>enable
node#configure
node(cfg)#copy tftp://172.16.55.1/cert1 pki:own-certificate/cert2
```

Example: Erase the two certificates

```
node>enable
node#configure
node(cfg)#erase pki:own-certificate/cert2
node(cfg)#erase pki:own-certificate/cert1
```

Trusted-certificate handling

Root certificates are certificates that are trusted by the device. If the peer presents a certificate that does inherit from one of them, the connection-handshaking is accepted.

Mode: Administrator exec

Step	Command	Purpose
1	<code>node(cfg)#export pki:trusted-certificate/name</code>	Displays a trusted certificate in Base64 format, ready to be copied from the terminal for export.
2	<code>node(cfg)#show pki:trusted-certificate/name</code>	Shows a trusted certificate logical content.
3	<code>node(cfg)#copy pki:trusted-certificate/name tftp://node-ip-address/filename</code>	Exports a trusted certificate to a TFTP server.
4	<code>node(cfg)#copy tftp://node-ip-address/filename pki:trusted-certificate/name</code>	Imports a trusted certificate from a TFTP server. A content check will be applied: If the file is not a valid certificate then it will not be imported.
5	<code>node(cfg)#erase pki:trusted-certificate/name</code>	Erases a trusted certificate.

Example: Imports a trusted certificate from a TFTP server

```
node>enable
node#configure
node(cfg)#copy tftp://172.16.55.1/cert1 pki:trusted-certificate/cert1
```


Example: Exports a trusted certificate to a TFTP server

```
node>enable
node#configure
node(cfg)#copy pki:trusted-certificate/cert1 tftp://172.16.55.1/cert2
```

Example: Display a trusted certificate in Base64 format

```
node>enable node#configure
node(cfg)#export pki:trusted-certificate/cert1
-----BEGIN CERTIFICATE-----
MIICNzCCAaACCQDSauxdaSaSLTANBgkqhkiG9w0BAQUFADBGMQswCQYDVQQGEwJD
SDENMASGALUECAwEgMmVyb2JENMASGALUEBwwEgMmVyb2JEMAwGALUECgwFSW5hbHAX
DDAKBgNVBAsMA1JORDEVMBMGALUEAwMMTCyLjE2LjU1LjQxMB4XDTEzMDcwMjA4
MzQxOFoXDTE0MDcwMjA4MzQxOFowYDELMAkGALUEBhMCQ0gxDTALBgNVBAGMBEJl
cm4xDTALBgNVBACMBEJlcm4xDjAMBGNVBAoMBUluYWxwMQwwCgYDVQQLDANSTkQx
FTATBgNVBAMMDDE3Mi4xNi41NS40MTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkC
gYEA697iz7j27FT19JkjrldzoPnFmYX/fTPzrGXLLaWasQJUat6xEYrngt0kgw
KtpXxAvt14HLP7j/530y9CET87w6bgcu3trh1J8jSQ+B7Yqx9TjuvvgGtMExxb5Q
rb/zMmtaKu7S7pQ8fijVYyOtWCyhsI5H6H0vQPGyLOeovsCAwEAATANBgkqhkiG
9w0BAQUFAAOBgQB7KvsIbiJmzoEQvZaZDlefbcDPIZdsAshcLwP7jE9bGC1knbv
aAcyB6n1Bt1lUQXDqbF0GhcauEEKBA3O7IlnjWyeebg58j5iKq89FGCaH1sQhXKO
z61A8YPl2gHYQcxCrZX7g9aQ9hwCc2OG+Mg+h4wpXbiOof2qM/3muk1FGQ==
-----END CERTIFICATE-----
```

Example: Shows the logical content of a trusted certificate

```
node>enable node#configure
node(cfg)#show pki:trusted-certificate/cert1
Certificate:
Data:
Version: 1 (0x0)
Serial Number:
d2:6a:ec:5d:69:26:92:2d
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=CH, ST=Bern, L=Bern, O=Inalp, OU=RND, CN=172.16.55.41
Validity
Not Before: Jul 2 08:34:18 2013 GMT
Not After : Jul 2 08:34:18 2014 GMT
Subject: C=CH, ST=Bern, L=Bern, O=Inalp, OU=RND, CN=172.16.55.41
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
Public-Key: (1024 bit)
Modulus:
00:9f:ee:bd:ee:2c:fb:8f:6e:c5:4f:5f:49:92:3a:
f5:77:3a:0f:9c:59:98:5f:f7:d3:3f:3a:c6:5c:b2:
da:59:ab:10:25:46:ad:eb:11:18:ae:78:2d:d2:48:
30:2a:da:57:c4:0b:ed:d7:81:cb:3f:b8:ff:e7:7d:
32:f4:21:13:f3:bc:3a:6e:07:2e:de:da:e1:d4:9f:
23:49:0f:81:ed:8a:b1:f5:38:ee:be:08:06:b4:c1:
31:5d:be:50:ad:bf:f3:32:6b:5a:2a:ee:d2:ee:94:
3c:7e:28:d5:63:23:ad:58:2c:a1:b0:8e:47:e8:73:
af:40:f1:b2:2c:e1:28:7a:fb
Exponent: 65537 (0x10001)
Signature Algorithm: sha1WithRSAEncryption
4f:ec:ab:ec:21:b8:89:9b:3a:04:42:f6:5a:64:39:5e:7d:b0:
```

```
9d:3c:86:5d:b0:0b:21:70:bc:0f:ee:31:3d:6c:60:b5:92:76:
ef:68:07:32:07:a9:f5:06:dd:65:51:05:c3:a9:b1:74:1a:17:
1a:b8:41:0a:04:0d:ce:ec:8d:67:8d:6c:9e:79:b8:39:f2:3e:
62:2a:af:3d:14:60:9a:1f:5b:10:85:72:8e:cf:ad:40:f1:83:
e5:da:01:d8:41:cc:42:ad:95:fb:83:d6:90:f6:1c:02:73:63:
86:f8:c8:3e:87:8c:29:c5:b8:8e:a1:fd:aa:33:fd:e6:ba:4d:
45:19
node>enable node#configure
node(cfg)#erase pki:trusted-certificate/cert1
```

Generated default files

In order to make the Private Key Infrastructure easier to use a couple of default files will be generated at the first startup. The files are generated only once, are immutable, and can be used in basic scenarios (see chapter 56, “[Secure SIP Applications](#)” on page 725, for example). The following default files are being generated:

- Private And public key
 - Name: DEFAULT
 - Key length: 512
- Own certificate
 - Name: DEFAULT
 - Self-signed with private key DEFAULT
 - Validity: 1000 years

Chapter 36 **Quality of Service (QoS) Overview**

Chapter contents

Introduction.....	424
Packet Classification.....	424
Type-of-Service (TOS)/Class-of-Service (CoS) Mapping.....	425

Introduction

This chapter provides an overview of the core principles of Patton's Quality-of-Service architecture. This chapter includes the following sections:

- “Packet Classification” on page 424
- “Type-of-Service (TOS)/Class-of-Service (CoS) Mapping” on page 425

QoS in networking refers to the capability of the network to provide a better service to selected network traffic. In the context of VoIP, the primary issue is to control the coexistence of voice and data packets such that voice packets are delayed as little as possible.

Currently, Patton devices do not provide a means of scheduling voice packets differently within Trinity. However, Trinity is able to convert external QoS-tags (IP TOS header or 802.1pq CoS header fields) between networks of different QoS realms. (Those tags may have a different meaning in different QoS realms.)

Packet Classification

Several Trinity services, such as access control lists and policy routing, need to distinguish between different types of packets. We refer to those types as “traffic classes”. You can think of the traffic-class as if every packet in the Trinity device has a tag attached to it on which the classification can be noted. Trinity's classifier can be used to apply such a traffic-class tag to some types of packets based on their protocol headers (e.g. source/destination address and port, packet length, IP TOS field etc.). Thus conceptually, as depicted in [figure 63](#), the classifier groups packet flows from different originating hosts but of the same service (e.g. data, voice, etc.) into virtual traffic class flows.

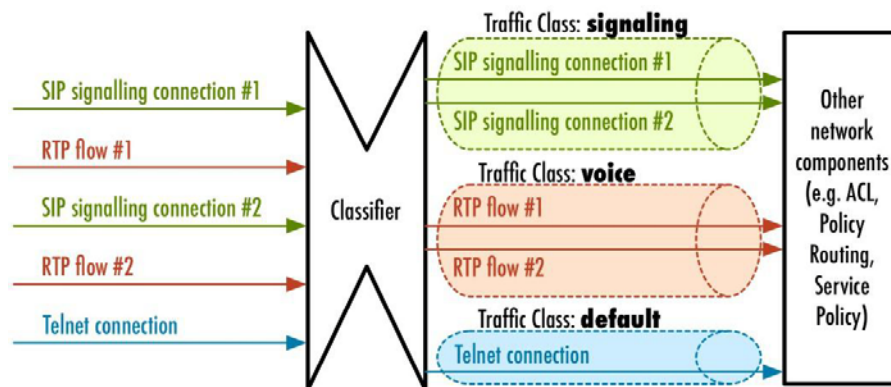


Figure 63. Conceptual view on the classifier; it groups packet flows of the same service

If not configured otherwise, the traffic-class tag is set to *DEFAULT* for all received packets. In contrast, locally-generated voice and data packets are tagged with the traffic class *LOCAL-VOICE* and *LOCAL-DEFAULT*, respectively.

Type-of-Service (TOS)/Class-of-Service (CoS) Mapping

The traffic-class tags exist only inside a Trinity device, but layer 2 priority bits (802.1pq class-of-service) and IP header type-of-service bits (TOS field) in received packets can be used to mark specific packet types for the other network nodes in a QoS-enabled network. The figure below shows that several Trinity services are involved in mapping TOS and CoS header fields to traffic classes and vice versa while a packet is routed through the device.

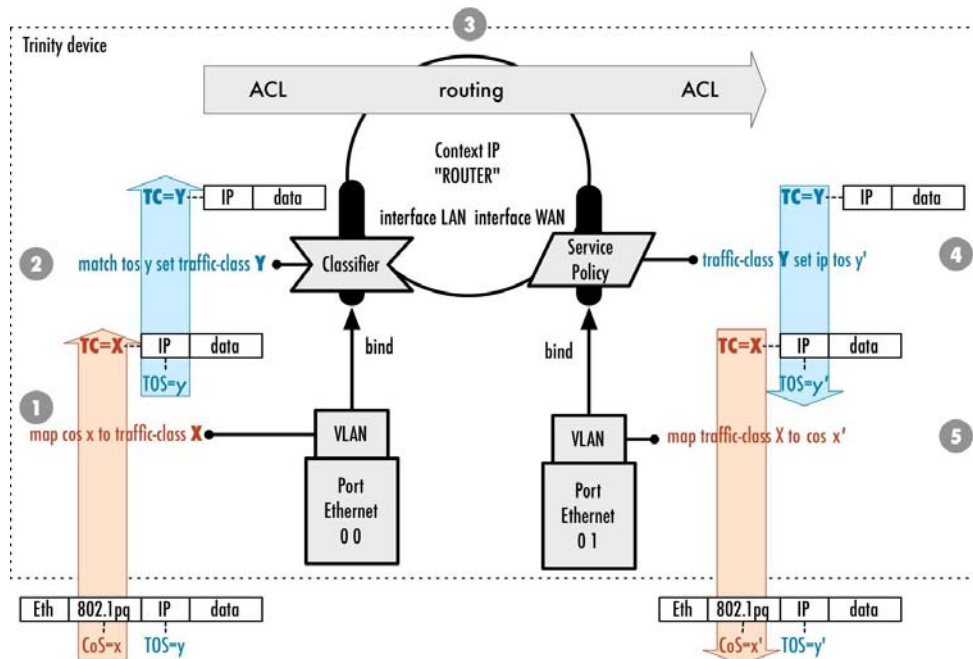


Figure 64. Mapping TOS/CoS to traffic-class and vice-versa

1. The **map** command on a VLAN port can be used to map an 802.1pq class-of-service value (CoS field) to an internal traffic class. This command is explained in more detail in Section “Configuring a VLAN” on page 201 in Chapter 14, “Ethernet Port Configuration” on page 197.
2. The **bind** command on a VLAN port determines over which IP interface a packet reaches Trinity’s IP router. The classifier that is attached to an IP interface can be used to map an IP type-of-service value (TOS field) to an internal traffic class. Note that this traffic class overwrites the traffic-class set on the VLAN port (if any). Consult Chapter 39, “Classifier Configuration” on page 450 for more information about how to set up a classifier.
3. Several network services such as Access Control Lists (ACL) or policy routing may use the internal traffic-class tag to treat packets of different services differently. The ACL, for example, may drop all packets belonging to a certain traffic class; the **route** command on an IP interface can be used to select a different routing table for other traffic classes. See Chapter 38, “Access Control List Configuration” on page 440 or Chapter 23, “IP Routing” on page 300 to learn how to configure these services.
4. After the router has found the IP interface over which the packet has to be sent, the packet traverses multiple services that may again inspect and manipulate the packet (see Chapter 21, “IP Context Overview” on page 280 for an overview). The service-policy profile is one of those services. It can be used to map an

internal traffic-class back to an IP type-of-service value (TOS field) (see 40, “[Service Policy Configuration](#)” on page 458 for more details).

5. Finally, if the packet leaves the device over a VLAN port, the **map** command may be used to convert the internal traffic class to a 802.1pg class-of-service value (CoS field) (see Chapter 14, “[Ethernet Port Configuration](#)” on page 197).

Chapter 37 Profile Service-Policy Configuration

Chapter contents

Introduction	428
Applying Scheduling at the Bottleneck	428
Using Traffic Classes	428
Patton DownStreamQoS™	428
Introduction to Scheduling	429
Priority	429
Weighted fair queuing (WFQ)	429
Shaping	429
Handling of bursts	429
Hierarchy	430
Quick References.....	430
Setting the Modem Rate	430
Configure DownStreamQoS™	430
Configuration Example: Common application case	431
Service-Policy configuration task list.....	432
Creating a service-policy profile	432
Configure Link arbiter	433
Rate limit	433
Arbiter mode	433
Source traffic-class	434
Hierarchical profile service-policy	434
Share for weighted-fair-queuing	434
Bit-rate for shaper	435
Assigning absolute priority	435
Real time traffic	435
Queue length	435
Queue type	436
Discarding excess load	436
Set QoS-related IP header field	436
Binding a classifier profile to the outbound traffic of an IP interface	437
Troubleshooting.....	438

Introduction

This chapter describes how to use and configure service-policy profiles. Service-policy profiles can be bound to individual IP interfaces to map internal traffic classes to QoS-related IP header fields. For an overview of Trinity's Quality-of-Service architecture and the meaning of traffic classes, refer to Chapter 36, "Quality of Service (QoS) Overview" on page 423.

This chapter describes how to use and configure the Quality of Service (QoS) features. This chapter includes the following sections:

- Quick references (see [page 430](#))
- Assigning bandwidth to traffic classes **[WHERE IS THIS CONTENT?]**
- Service Policy configuration task list (see [page 432](#))

QoS in networking refers to the capability of the network to provide a better service to selected network traffic. In the context of VoIP, the primary issue is to control the coexistence of voice and data packets such that voice packets are delayed as little as possible. This chapter shows you how to configure Trinity to best use the access link.

In many applications you can gain a lot by applying the minimal configuration found in the quick reference section, but read sections "Applying Scheduling at the Bottleneck" and "Using Traffic Classes" first to understand the paradox of why we apply a rate-limit to reduce delay and what a *traffic-class* means.

Applying Scheduling at the Bottleneck

When a Patton device acts as an access router and voice gateway, sending voice and data packets to the Internet, the access link is the point where intelligent use of scarce resources really makes a difference. Frequently, the access link modem is outside of the Patton device and the queuing would happen in the modem, which does not distinguish between voice and data packets. To improve QoS, you can configure the Patton device to send no more data to the Internet than the modem can carry. This keeps the modem's queue empty and gives the Patton device control over which packet is sent over the access link at what time.

Using Traffic Classes

The Service Policy needs to distinguish between different types of packets. We refer to those types as *traffic-classes*. You can think of the traffic-class as if every packet in the Patton device has a tag attached to it on which the classification can be noted. The classifier can be used to apply such a traffic-class name to some type of packet based on its IP-header filtering capabilities. The traffic-class tags exist only inside the Patton device, but layer 2 priority bits (802.1pq class-of-service) and IP header type-of-service bits (TOS field) can be used to mark a specific packet type for the other network nodes. By default the traffic-class tag is empty. Only two types of packets are automatically marked by the Software: voice packets and data packets originated from the Patton device itself are marked as *local-voice* and *local-default* respectively.

Patton DownStreamQoS™

For traffic flowing downstream from the Internet, traditional access routers cannot control the volume nor the sending users. Although downstream traffic is usually requested and initiated by users on the local network (LAN), controlling these upstream requests is not sufficient to limit the downstream traffic effectively because one request may trigger a server to send an entire file over the downstream path. The ISP's edge router commonly responds to a packet rate that exceeds the link bandwidth by discarding VoIP packets with the same probability as any other packet type. These factors, or a combination of them, may degrade voice quality to a

degree that users find objectionable. Unlike data traffic -which can be retransmitted- real-time voice traffic is vulnerable to packet loss since it leads to audible interruption.

To resolve the problem of degraded voice quality for incoming traffic, Patton has devised a leading-edge technology called DownStreamQoS™. Within the Patton device deployed at the customer premise, DownStreamQoS dynamically creates a virtual bottleneck against the incoming packet stream. This bottleneck can throttle non-real-time traffic, preventing the edge router from blocking or impeding voice traffic, to ensure voice or other real-time packets are transmitted freely downstream. DownStreamQoS exploits flow-control mechanisms within the TCP standard to create the bottleneck. Because 80% of Internet traffic is transported via TCP, DownStreamQoS is especially effective. See section “[Configure DownStreamQoS™](#)” on page 430.

Introduction to Scheduling

Scheduling essentially means to determine the order in which packets of the different traffic-classes are served. The following sections describe the ways this arbitration can be done.

Priority

One way of ordering packets is to give priority to one traffic-class and to serve the other traffic-classes when the first has nothing to send. Trinity uses the priority scheme to make sure that voice packets generated by the Patton device will experience as little delay as possible. Voice packets can receive this treatment because they will not use up the entire bandwidth.

Weighted fair queuing (WFQ)

This arbitration method assures a given minimal bandwidth for each source. An example: you specify that traffic-class A gets three times the bandwidth of traffic-class B. So A will get a minimum of 75% and B will get a minimum of 25% of the bandwidth. But if no class A packets are waiting B will get 100% of the bandwidth. Each traffic-class is in fact assigned a *relative* weight, which is used to share the bandwidth among the currently active classes. Patton recommends that you specify the weight as percent which is best readable.

Shaping

There is another commonly used way to assign bandwidth. It is called *shaping* and it makes sure that each traffic-class will get just as much bandwidth as configured and not more. This is useful if you have subscribed to a service that is only available for a limited bandwidth e.g. low delay. When connecting the Patton device to a *Diff-Serv* network shaping might be a required operation.

Handling of bursts

For shaping there is a variation of the scheduler that allows to specify if a traffic class may temporarily receive a higher rate as long as the average stays below the limit. This burstiness measure allows the network to explicitly assign buffers to bursty sources. When you use shaping on the access link the shaper sometimes has the problem that multiple sources are scheduled for the same time - and therefore some of them will be served too late. If the rate of every source had to strictly obey its limit, all following packets would also have to be delayed by the same amount, and further collisions would reduce the achieved rate even further. To avoid this effect, the Trinity shaper assumes that the burstiness needed for sources to catch up after *collisions* is implicitly allowed. Future versions of Trinity might allow setting the burst rate and bursting size if more control over its behavior is considered necessary..

Hierarchy

An arbiter can either use *wfq* or *shaping* to determine which source to serve next. If you want the scheduler to follow a combination of decision criteria you can combine different schedulers in hierarchy to do a multi-level arbitration. Hierarchical scheduling is supported in Trinity with service-policy profiles used inside service policy profiles. In Figure 23 **[MISSING FIGURE?]** an example of hierarchical scheduling is illustrated. The 1st level arbiter *Level_1* uses weighted fair queuing to share the bandwidth among source classes VPN, Web and incorporates the traffic from the 2nd level arbiter *Low_Priority*, which itself uses shaping to share the bandwidth among source classes Mail and Default.

Quick References

The following sections provide a minimal “standard” service-policy configuration for the case where voice and data share a (DSL/cable) modem link.

Setting the Modem Rate

To match the voice and data multiplexing to the capacity of the access link is the most common application of the Trinity service-policy.

1. Create a minimal profile.

```
profile service-policy MODEM-512
  rate-limit 512 overhead 30 atm
  source traffic-class local-voice
  priority
```

2. Apply the profile just created to the interface connected to the modem.

```
context ip
  interface WAN
    use profile service-policy MODEM-512 out
```

Some explanations:

- **MODEM-512** is the title of the profile which is referred to when installing the scheduler
- **rate-limit 512** allows no more than 512 kbit/sec to pass which avoids queuing in the modem.
- **overhead 30** specifies how many framing bytes are added by the modem to "pack" the IP packet on the link. The framing is taken into account by the rate limiter.
- **atm** tells the rate limiter that the access link is ATM based. This option includes the ATM overhead into the rate limit calculation. Please add 8 bytes to the overhead for AAL5 in this case.
- **source traffic-class** enters a sub-mode where the specific handling for a traffic-class is described. The list of sources in the service-policy profile tells the arbiter which "traffic sources" to serve.
- **local-voice** is the predefined traffic-class for locally terminated voice packet streams.
- **priority** means that packet of the source being described are always passed on immediately, packets of other classes follow later if the rate limit permits.

Configure DownStreamQoS™

The Patton device can throttle downstream data traffic on the access link, minimizing the risk of lost or delayed voice or other real-time packets coming from the Internet or WAN.

The example below defines a service-policy profile applied to incoming traffic on the WAN interface:

```
profile service-policy DOWN-LINK
  rate-limit 768 voice-margin 200

  source traffic-class local-voice
    priority
    real-time

  source traffic-class voice
    priority
    real-time

  source traffic-class default
    queue-limit 4

Interface WAN
  use profile service-policy in DOWN-LINK
```

The command parameters in the example above function as described below:

- **voice-margin:** For DownStreamQoS (only for incoming traffic) Trinity reduces the maximum bandwidth by the specified bit rate in kbit/sec (200 kbit/sec in this example). The throttling effectively reduces the average rate of incoming *data* packets, which frees bandwidth for priority traffic thus improving voice quality. For best results, use a generous value for voice-margin to accommodate fluctuations in traffic flow. 33% of the total bandwidth is recommended, up to a maximum of 200 kbit/sec (this default behavior can be easily configured with the voice-margin *auto* option).
- **priority:** Means that packet of the local-voice source are always passed on immediately, packets of other classes follow later if the rate limit permits.
- **real-time:** If at least one of the listed traffic classes contains the *real-time* command the service policy filters-out bursts in the data traffic to keep the delay of voice streams small and without fluctuations.

Note This command makes only sense if used along with the *priority* command.

- **queue-limit:** This setting defines how many packets are to be enqueued before dropping excess traffic (data packets). A dropped packet wastes bandwidth but it also tells the sender to slow down (reduce its transmission rate) because the link is overloaded. The default queue size in Trinity is 16. By reducing the queue limit, the Patton device throttles data more effectively, further improving voice quality.
- **source traffic-class local-voice:** Incoming and outgoing local RTP traffic is automatically tagged as *local-voice*. This default traffic class can be used to prioritize local RTP traffic.
- **source traffic-class voice:** Incoming UDP traffic, that is not local with a length smaller than 200 Bytes and with the first two payload bits equal to the RTP version 2, is tagged as voice. This default traffic class can be used to prioritize incoming routed RTP traffic.

Note Outgoing routed RTP traffic is not automatically tagged.

Configuration Example: Common application case

If the default traffic-class *voice* and *local-voice* are not sufficient for prioritizing the voice traffic, the user can use a *profile classifier* as in the following example. This classifier profile tags voice packets in LAN and WAN inter-

faces as traffic-class VOICE and a service-policy profile which sets priorities and rate limitation on all traffic (in/out) in WAN interface:

```

profile classifier VOICE_CLASSIFIER
  match 1 length 40..256 protocol udp dest-port 5000..10000 set traffic-class VOICE

profile service-policy WAN_SVC_POL
  rate-limit 768 voice-margin 200

  source traffic-class VOICE
    priority
    real-time

  source traffic-class local-voice
    priority
    real-time

  source traffic-class default
    queue-limit 4

context ip ROUTER

interface LAN
  use profile classifier in VOICE_CLASSIFIER

interface WAN
  use profile classifier in VOICE_CLASSIFIER
  use profile service-policy in WAN_SVC_POL
  use profile service-policy out WAN_SVC_POL

```

Service-Policy configuration task list

To configure service-policy profiles, perform the tasks in the following sections:

- Creating a service-policy profile (see [page 432](#))
- Configure link arbiter (see [page 433](#))
- Set QoS-related IP header field (see [page 436](#))
- Binding a classifier profile to the outbound traffic of an IP interface (see [page 437](#))

Creating a service-policy profile

The service-policy profile defines whether and how to set IP header fields such as TOS, precedence, DSCP, and ECN based on the traffic-class tag of an outgoing packet.

This procedure describes how to create a service-policy profile and enter the profile's configuration mode.

Mode: administrator exec

Step	Command	Purpose
1	<i>node(cfg)#profile service-policy</i> <i>pfname</i>	Creates the service-policy profile <i>pfname</i> and enters its configuration mode.

The parameter *pfname* is the identifier by which the profile will be known. Entering this command puts you into service-policy profile configuration mode where you can enter traffic-class-specific modes as shown in the next section.

Use the **no** form of this command to delete a service-policy profile. You cannot delete a service-policy profile if it is currently bound to an interface. When you modify a service-policy profile, the new settings immediately become active.

Example: Create a service-policy profile

In the following example the service-policy profile named *SP_WAN* is created.

```
node>enable
node#configure
node(cfg)#profile service-policy SP_WAN
node(pf-svcpol)[SP_WAN]#
```

Configure Link arbiter

For the link arbitration the profile service-policy holds all the necessary parameters.

Rate limit

For QoS it is important to be at the bottleneck and control the queue of the bottleneck in order to control which streams get which share and which packets needs to be dropped in the case of an overload of the link. Therefore the rate-limit needs to be configured to a value slightly smaller than the actual available link.

Mode: Profile service-policy

Step	Command	Purpose
1	<i>node(svcpol-clis)[pfname]#[no] rate-limit<kbits> [overhead <Bytes>] [ethernet atm][voice-margin <kbits> auto]</i>	Limits global interface rate to value in kbits. The configured value should be slightly smaller than the real available uplink. Overhead specifies if there are additional header bytes added from the modem, which are more than the normal Ethernet header. With link-layer atm the padding for atm cells can be taken into account. Voice-margin is subtracted from total limit for DownstreamQoS (virtual bottleneck). Default: disabled. Default link-layer: Ethernet. Default overhead: 0 Bytes. Default voice-margin: 0.

Arbiter mode

The arbiter mode defines if the distribution of bandwidth to traffic-classes or hierarchical policies happens with limiting to maximum values (shaper) or guaranteeing minimum values (wfq). For achieving a mixture of both arbitration modes one needs to define a hierarchical service-policy profile and include that as source in another service policy profile.

Mode: Profile service-policy

Step	Command	Purpose
1	node(svcpol-cl) [<i>pfname</i>] #mode (shaper wfq)	Set the arbiter mode to wait-fair-queuing (wfq) or shaper. Default: wfq

Source traffic-class

The main sources of packets for the arbiter are packets belonging to traffic-classes. The packets get the traffic-class form the classifier. In the arbiter the share or bandwidth is assigned to traffic classes. Packets belonging to a traffic-class not explicitly handled in the profile service-policy get only scheduled if there is enough left bandwidth after handling all the other traffic-classes.

Mode: Profile service-policy

Step	Command	Purpose
1	node(svcpol-cl) [<i>pfname</i>] #[no] source traffic-class <traffic-class>	Enters the configuration mode for a traffic-class.

Hierarchical profile service-policy

It is possible to have hierarchical service policy profiles as a source of other service policy profiles. This can be used for achieving a mixture of the arbiter modes shaper and wfq. The other use-case is setting up the borrowing between traffic-classes in the wfq mode. Borrowing means that bandwidth of a traffic-class which is not used is distributed even for all the sources in the same hierarchical profile. An only after that it is given to other traffic-classes in other profiles.

Mode: Profile service-policy

Step	Command	Purpose
1	node(svcpol-cl) [<i>pfname</i>] #[no] sourcepolicy <service-policy profile>	Enters the configuration mode of a hierarchical lower level service-policy profile.

Share for weighted-fair-queuing

The command **share** is used with wfq link arbitration to assign the weight to the selected traffic-class. When defining a number of source classes, the values are relative to each other. It is recommended to split 100—which can be read as 100%—among all available source classes, e.g. with 20, 30 and 50 as value for the respective share commands, which represent 20%, 30% and 50%.

This command is only used when the arbiter is in wfq mode. For the shaper mode it is ignored. Traffic-classes without share get no guaranteed bandwidth and therefore are sent only if there is some bandwidth left from the others.

Mode: Source traffic-class and policy

Step	Command	Purpose
1	node(src) [<i>name</i>] #[no] share <percentage>	Defines fair queuing weight for minimal guaranteed bandwidth (relative to other sources) to percentage for the selected class or policy name. Default: disabled

Bit-rate for shaper

The command **rate** is used with shaper link arbitration to assign the (average) bit-rate to the selected source. When enough bandwidth is available each source will exactly receive this bandwidth (but no more), when over-loaded the shaper will behave like a wfq arbiter. Bit-rate specification for shaper (kilobits).achieving a mixture of both arbitration modes one needs to define a hierarchical service-policy profile and include that as source in another service policy profile.

This command is only used when the arbiter is in shaper mode. For the wfq mode it is ignored. Traffic-classes without rate are degraded to a lower priority and get served only if all other traffic-classes are served before.

Mode: Source traffic-class and policy

Step	Command	Purpose
1	node(src)[name]#[no] rate <kbits>	Defines the (average) bit-rate to the selected in kbps <i>kilo-bits</i> or as <i>remaining</i> if a second priority source is getting the unused bandwidth for the selected class or policy <i>name</i>

Assigning absolute priority

This command **priority** can only be applied to classes, but not to lower level polices. The class is given absolute priority effectively bypassing the link arbiter. Care should be taken, as traffic of this class may block all other traffic. The packets given “priority” are taken into account by the “rate-limit”. Use the command **police** to control the amount of “priority” traffic.

Mode: Source traffic-class

Step	Command	Purpose
1	node(src)[name]#[no] priority	Defines absolute priority effectively bypassing the link arbiter for the selected class or policy <i>name</i>

Real time traffic

The command **real-time** specifies that the traffic in this class requires a completely fixed rate with minimum delay for user interactivity (i.e. voice). This command minimizes packet-bursts processing so no real-time packets are being delayed after them, keeping the rate as fix as possible.

Note This command doesn't prioritize the traffic, use the priority command prioritization.

Mode: Source traffic-class

Step	Command	Purpose
1	node(src)[name]#[no] real-time	Configures the traffic as real-time, optimizing it for minimum-delay

Queue length

The command **queue-limit** specifies the maximum number of packets queued for the class *name*. Excess packets are dropped. Used in “class” mode—queuing only happens at the leaf of the arbitration hierarchy tree. The

no form of this command reverts the queue-limit to the internal default value, which depends on your configuration.

Mode: Source traffic-class

Step	Command	Purpose
1	node(src)[name]#[no] queue-limit <packets>	Defines the maximum number of packets queued for the selected class

Queue type

The type of queue defines the drop-behavior and/or has an influence to the fairness between streams of the same traffic-class.

Mode: Source traffic-class

Step	Command	Purpose
1	node(src)[name]#queue-mode fifo	The packets first come are first served. When the queue is full, excess packets are dropped (taildrop). This is the default setting.
1	node(src)[name]#queue-mode sfq [flows <number of flows>]	The streams are divided into sub-queues and these are served all with equal weight. Flows control the number of sub-queues. Default flows: 1024.
1	node(src)[name]#queue-mode red [burst-tolerance <1-10>]	Defines random early detection (RED) for queues of for the selected traffic-class or policy <i>name</i> . The range for the optional value <i>burst-tolerance</i> is from 1 to 10. Default burst-tolerance: 5.

Discarding excess load

The command **police** controls traffic arriving in a queue for class *name*. The value of the first argument *average-kilobits* defines the average permitted rate in kbps, the value of the second argument *kilobits-ahead* defines the tolerated burst size in kbps ahead of schedule. Excess packets are dropped.

Mode: Source traffic-class

Step	Command	Purpose
1	node(src)[name]#[no] police <averagerate> burst-size <kbits-ahead>	Defines how traffic arriving in a queue for the selected class or policy <i>name</i> has to be controlled. The value <i>average-kilobits</i> for average rate permitted is in the range from 0 to 10000 kbps. The value <i>kilobits-ahead</i> for burst size tolerated ahead of schedule is in the range from 0 to 10000.

Set QoS-related IP header field

When a service-policy profile is bound to an IP interface, the profile may set IP header fields such as TOS, precedence, DSCP, and ECN of all packets sent over that IP interface. For each internal traffic class, a separate set of header field values can be configured.

This procedure describes how to map an internal traffic class to a set of IP header fields.

Mode: Profile service-policy

Step	Command	Purpose
1	node(svcpol-cl) [<i>pfname</i>] #source traffic-class <i>traffic-class</i>	Enters a sub-mode that combines configuration commands for packets tagged with the specified <i>traffic-class</i> .
2a	node(src-tc) [<i>traffic-class</i>] #set tos <i>tos</i>	Sets the IP type-of-service header field to <i>tos</i> .
3a	node(src-tc) [<i>traffic-class</i>] #set precedence <i>precedence</i>	Sets the IP precedence header field to <i>precedence</i> .
2b	node(src-tc) [<i>traffic-class</i>] #set dscp <i>dscp</i>	Sets the IP differentiated-services-code-point header field to <i>dscp</i> .
3b	node(src-tc) [<i>traffic-class</i>] #set ecn <i>ecn</i>	Sets the IP explicit-congestion-notification field to <i>ecn</i> .

Note The TOS/precedence header fields use the same bits as the DSCP/ECN fields. Thus you can either configure TOS and/or precedence values or DSCP and/or ECN values. If you for example try to enter a DSCP value after having configured a TOS value, the DSCP value is not accepted and you will be presented with an error message.

Example: Set IP TOS and precedence fields

The following example prepares a service-policy profile to set the IP TOS field to 5 for all packets tagged with the *MGMT* traffic class.

Note The profile must be bound to an IP interface in order to get activated (see section below).

```
node(cfg)#profile service-policy SP_WAN
node(pf-svcpol)[SP_WAN]#source traffic-class MGMT
node(src-tc)[MGMT]#set tos 5
```

Binding a classifier profile to the outbound traffic of an IP interface

To apply a service-policy profile to the outbound traffic it has to be bound to an IP interface. This procedure describes how to do so with the **use** command.

Mode: IP interface

Step	Command	Purpose
1	node(if-ip) [<i>ifname</i>] #use profile service-policy [<i>in out</i>] <i>pfname</i>	Binds the service-policy profile <i>pfname</i> to outgoing packets on the IP interface <i>ifname</i> .

The command offers the following options:

Parameter	Explanation
<i>ifname</i>	Name of the IP interface to which a service-policy profile gets bound.

Parameter	Explanation
<i>pname</i>	The name of a service-policy profile that has already been created using the profile service-policy command. This argument must be omitted in the no form.
in	Specifies that the service-policy profile applies to packets received over this interface. This currently is of limited use as packet tagging should always happen at an egress point of the device to match the QoS-field semantics of the target network.
out	Specifies that the service-policy profile applies to packets sent over this interface.

The **no** form of the **use** command is used to unbind a service-policy profile from an IP interface. When using this form, you don't have to specify the profile name.

Mode: IP Interface

Step	Command	Purpose
1	node(if-ip)[ifname]#no use profile service-policy [in out]	Unbinds the service-policy profile for incoming or outgoing packets from the IP interface <i>ifname</i> .

Parameter	Explanation
<i>ifname</i>	Name of the IP interface on which a service-policy profile gets unbound.
in	Specifies that the service-policy profile applies to packets received over this interface.
out	Specifies that the service-policy profile applies to packets sent over this interface.

Example: Bind and unbind a service-policy profile.

Bind the service-policy profile created in the previous example to tag outgoing packets on interface *WAN* in the default IP context (*ROUTER*).

```
node(cfg)#context ip ROUTER
node(ctx-ip)[ROUTER]#interface WAN
node(ip-if)[ROUTER.WAN]#use profile service-policy out SP_WAN
```

Unbind the SP_WAN service-policy profile from outgoing traffic on interface WAN:

```
node(ip-if)[ROUTER.WAN]#no use profile service-policy out
```

Troubleshooting

In the case of an Issue with the service-policy profile the first thing to check is if the profile could be applied to the system successfully. Execute on a newly booted device:

- Show log boot
- Show log error

If there are errors logged for service-policy then you should open an incident at your Patton support representative with the log files and your startup-configuration. For the meantime a nearing approach can be tried:

- Make sure the underlying transport (in most cases Ethernet) is up and running.
- Make sure the underlying transport (in most cases Ethernet) is bound to the ip interface.

- Unbind the failing profile service-policy from the ip interface.
- Create a new and empty profile service-policy.
- Bind the new profile service-policy to the ip interface.
- Enter the new profile service-policy and execute one command at once until it is rejected with an error or the final desired configuration is reached.
- When there is a command rejected due to failure, you may try to change parameters, use alternative commands or omit the command for going forward.

Chapter 38 Access Control List Configuration

Chapter contents

Introduction	441
About Access Control Lists (ACLs).....	441
What Access Lists Do	441
Why You Should Configure Access Lists	442
When to Configure Access Lists	442
Features of Access Control Lists	443
Access Control List Configuration Task List.....	443
Mapping Out the Goals of the Access Control List	443
Creating an Access Control List Profile and Enter Configuration Mode	444
Adding and Deleting a Filter Rule to the Current Access Control List Profile	444
Binding and Unbinding an Access Control List Profile to an IP Interface	445
Displaying an Access Control List Profile	447
Examples	448
Denying a Specific Subnet	448
Denying Traffic Between Two Interfaces	449
Permit Only Traffic Generated From LAN	449

Introduction

This chapter provides an overview of IP Access Control Lists (ACLs) and describes the tasks involved in configuring them.

This chapter includes the following sections:

- About ACLs
- ACL configuration task list (see page 443)
- Examples (see page 448)

About Access Control Lists (ACLs)

This section briefly describes what access lists do, why and when you should configure access lists, and basic versus advanced access lists.

What Access Lists Do

ACLs implement a firewall by filtering network traffic, forwarding or dropping each packet based on the ACL rules and bindings. ACL rules may match packets by any of the criteria listed in chapter (see Chapter 40, “[Packet Matching](#)” on page 458). ACL bindings determine which ingress and/or egress interface will be matched.

The firewall is a stateful firewall, meaning that packets may be matched not only based on information that can be determined from that packet alone, e.g. ingress/egress interface or header information such as source/destination IP addresses, but also based on the connection state which is determined by examining previous packets. This is useful because many networking application protocols have an initial connection that can be matched with well-known criteria, e.g. the server's port number, but then use the initial connection to negotiate a second, related connection using dynamically assigned port numbers. For these applications to work through a stateless firewall, the entire port range possible for that related connection must be permitted, usually all ports 1024 and greater, which greatly decreases the security of the firewall. A stateful firewall, on the other hand, understands how the related connection is negotiated by the initial connection and dynamically permits that related connection once it is negotiated.

The router tracks the connection state for the following protocols:

- ICMP
- TCP
- UDP

and the following applications:

- SIP
- FTP
- TFTP
- PPTP
- SANE
- IRQ

Note Sophisticated users can sometimes successfully evade or fool basic access lists because no authentication is required.

Why You Should Configure Access Lists

You should use ACLs to provide a basic level of security for accessing your network. If you do not configure ACLs on your router, all packets passing through the router could be allowed onto all parts of your network.

For example, ACLs can allow one host to access a part of your network, and prevent another host from accessing the same area. In figure 65, host A is allowed to access the Human Resources network and host B is prevented from accessing the Human Resources network.

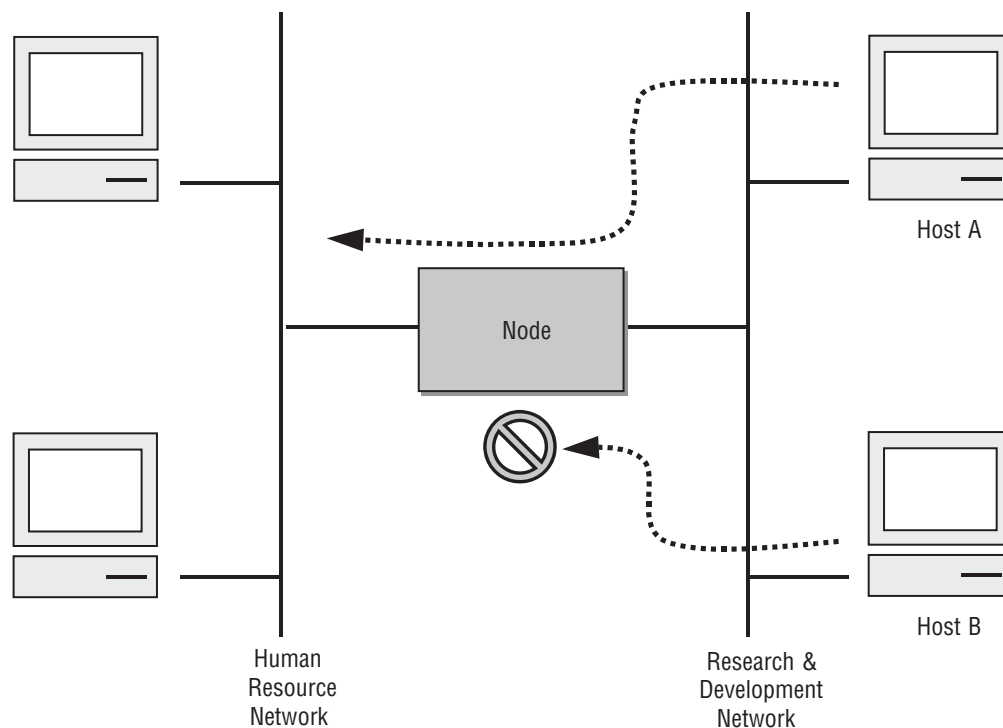


Figure 65. Using traffic filters to prevent traffic from being routed to a network

You can also use access lists to decide which types of traffic are forwarded or blocked at the device interfaces. For example, you can permit e-mail traffic to be forwarded but at the same time block all Telnet traffic.

When to Configure Access Lists

ACLs should be used in firewall routers, which are often positioned between your internal network and an external network such as the Internet. You can also use ACLs on a router positioned between two parts of your network, to control traffic entering or exiting a specific part of your internal network.

To provide the security benefits of ACLs, you should configure ACLs at least on border routers, i.e. those routers situated at the edges of your networks. This provides a basic buffer from the outside network or from a less controlled area of your own network into a more sensitive area of your network.

On these routers, you should configure ACLs for each network protocol configured on the router interfaces. You can configure ACLs so that inbound traffic or outbound traffic or both are filtered on an interface.

Features of Access Control Lists

The following features apply to all IP ACLs:

- An ACL may contain multiple rules. The order of rules is significant. Each rule is processed in the order it appears in the configuration file. As soon as a rule matches, the corresponding action is taken and no further processing takes place.
- An IP interface may be bound to multiple ACLs. The order in which the ACLs are bound is significant. Each ACL is scanned in the order in which it was bound. If no matching rule is found in the first ACL, the second ACL is scanned, and so on. If no matching rule is found in any of the ACLs, the packet is dropped. That is, there is an implicit **deny** all rule at the end of the last ACL.

Note Two or more administrators should not simultaneously edit the configuration file. This is especially the case with access lists. Doing this can have unpredictable results.

Once in access control list configuration mode, each command creates a statement in the access control list. When the access control list is applied, the action performed by each statement is one of the following:

- **permit** statement causes any packet matching the criteria to be accepted.
- **deny** statement causes any packet matching the criteria to be dropped.

Access Control List Configuration Task List

To configure an IP access control list, perform the tasks in the following sections.

- Mapping out the goals of the ACL
- Creating an ACL profile and enter configuration mode (see page 444)
- Adding a filter rule to the current ACL profile (see page 444)
- Binding and unbinding an ACL profile to an IP interface (see page 445)
- Displaying an ACL profile (see page 447)

Mapping Out the Goals of the Access Control List

To create an ACL, you must:

- Assign a unique name to the access list
- Define packet-filtering criteria

A single access control list can have multiple rules.

Before you begin to enter the commands that create and configure the ACL, be sure that you are clear about what you want to achieve with the firewall. Consider whether it is better to deny specific accesses and permit all others or to permit specific accesses and deny all others.

Note A single ACL can have multiple rules, but editing those rules online can be tedious. Therefore, we recommend editing complex ACLs offline within a configuration file and downloading the configuration file later via TFTP to your device.

Creating an Access Control List Profile and Enter Configuration Mode

This procedure describes how to create an ACL and enter ACL configuration mode.

Mode: Administrator execution

Step	Command	Purpose
1	device(cfg)#profile acl <i>name</i>	Creates the ACL profile <i>name</i> and enters the configuration mode for this ACL.

The ACL profile will be known by *name*. Entering this command puts you into *ACL configuration mode* where you can enter the individual statements that will make up the ACL rules.

Use the **no** form of this command to delete an ACL profile. You cannot delete an ACL profile if it is currently bound to an interface.

Example: Create an ACL profile

In the following example, the ACL profile named *WAN_RX* is created and the shell of the ACL configuration mode is activated.

```
device>enable
device#configure
device(cfg)#profile acl WAN_RX
device(pf-acl)[WAN_RX]#
```

Adding and Deleting a Filter Rule to the Current Access Control List Profile

The commands **permit** or **deny** are used to define an ACL rule. This procedure describes how to create an ACL rule.

Mode: ACL configuration

Step	Command	Purpose
1	device(pf-acl)device#{permit deny} <i>match</i>	Creates an ACL rule that either permits or denies access according to the <i>match</i> .

The table below explains the syntax:

Keyword	Meaning
permit	Forward any packet matching the <i>match</i> criteria.
deny	Drop any packet matching the <i>match</i> criteria.
match	See Chapter 40, “ Packet Matching ” on page 458.

If no *match* is specified, the rule will match all packets, so if you place the rule deny at the top of an access control list profile, no packets will pass regardless of the other rules you defined.

Example: Create ACL rules

Select the ACL profile named WAN_RX and create some rules to:

- drop all ICMP echo requests (as used by the ping command),
- but forward all other ICMP traffic,
- forward any TCP traffic to host 193.14.2.10 via port 80,
- forward UDP traffic from host 62.1.2.3 to host 193.14.2.11 via any port in the range from 1024 to 2048
- forward all traffic from the 97.123.121.0/24 subnet to host 193.14.2.11.

```
device(cfg)#profile acl WAN_RX
device(pf-acl)[WAN_RX]#deny protocol icmp icmp-type echo-request
device(pf-acl)[WAN_RX]#permit protocol icmp
device(pf-acl)[WAN_RX]#permit protocol tcp dest-ip 193.14.2.10 dest-port 80
device(pf-acl)[WAN_RX]#permit protocol udp src-ip 62.1.2.3 dest-ip 193.14.2.11
dest-port 1024..2048
device(pf-acl)[WAN_RX]#permit src-ip 97.123.121.0/24 dest-ip 193.14.2.11
device(pf-acl)[WAN_RX]#exit
device(cfg)#
```

The no form of the permit or deny command is used to delete an ACL rule. This procedure describes how to delete an ACL rule.

Mode: ACL Configuration

Step	Command	Purpose
1	device(pf-acl)device#show running-config current-mode	Show the ACL rules in the ACL <i>name</i> . Use this command to get the index of the rule you want to delete. You will use it in step 2.
2	device(pf-acl)device#no permit index	Removes the ACL rule at the specified <i>index</i> .

Example: Delete an ACL rule

Select the ACL profile named WAN_RX and delete the rule to permit any TCP traffic to host 193.14.2.10 via port 80.

```
device(cfg)#profile acl WAN_RX
device(pf-acl)[WAN_RX]#show running-config current-mode
deny 1 protocol icmp icmp-type echo-request
permit 2 protocol icmp
permit 3 protocol tcp dest-ip 193.14.2.10 dest-port 80
permit 4 protocol udp src-ip 62.1.2.3 dest-ip 193.14.2.11 dest-port 1024..2048
permit 5 src-ip 97.123.121.0/24 dest-ip 193.14.2.11
device(pf-acl)[WAN_RX]#no permit 3
device(pf-acl)[WAN_RX]#exit
device(cfg)#
```

Binding and Unbinding an Access Control List Profile to an IP Interface

The command **use** is used to bind an ACL profile to an IP interface. This procedure describes how to bind an ACL profile to incoming packets on an IP interface.

Mode: Interface

Step	Command	Purpose
1	device(if-ip)[if-name]#use profile acl in name [to { local interface dest-if-name}]	Binds ACL profile <i>name</i> to incoming packets on IP interface <i>if-name</i> . If local is specified, the ACL only applies to packets destined to the router itself. If <i>dest-if-name</i> is specified, the ACL only applies to packets destined to be routed out of <i>dest-if-name</i> . Otherwise, the ACL applies to all incoming packets on IP interface <i>if-name</i> regardless of the destination.

This procedure describes how to bind an ACL profile to all outgoing packets on an IP interface.

Mode: Interface

Step	Command	Purpose
1	device(if-ip)[if-name]#use profile acl out name	Binds ACL profile <i>name</i> to all outgoing packets on IP interface <i>if-name</i> .

The table below explains the syntax:

Keyword	Meaning
if-name	The name of the IP interface to which an ACL profile is bound.
name	The name of an ACL profile that has already been created using the profile acl command.
local	(Optional) If specified, the binding only applies to packets destined to the router itself.
dest-if-name	(Optional) If specified, the binding only applies to packets destined to be routed out of the IP interface by this name.

The no form of the use command is used to unbind an ACL profile from an IP interface. This procedure describes how to unbind an ACL profile from an IP interface.

Mode: Interface

Step	Command	Purpose
1	device(if-ip)[if-name]#show running-config current-mode	Shows the index to use to unbind the desired ACL.
2	device(if-ip)[if-name]#no use profile acl {in out} index	Unbinds ACL profile at <i>index</i> for either incoming or outgoing packets on IP interface <i>if-name</i> .

The table below explains the syntax:

Keyword	Meaning
if-name	The name of the IP interface from which the ACL profile is unbound.

Keyword	Meaning
in	Unbind the profile from incoming packets.
out	Unbind the profile from outgoing packets.
index	The index of the access control list which is unbound.

Example: Bind and unbind an ACL profile to/from an IP interface

Bind ACL profile WAN_RX to incoming packets on the interface *WAN* in the IP router context.

```
device(cfg)#context ip ROUTER
device(cfg-ip)[ROUTER]#interface WAN
device(cfg-if)[WAN]#use profile acl WAN_RX in
```

Bind ACL profile WAN_RX from incoming packets on the interface *WAN* in the IP router context.

```
device(cfg)#context ip ROUTER
device(cfg-ip)[ROUTER]#interface WAN
device(cfg-if)[WAN]#no use profile acl in 1
```

Displaying an Access Control List Profile

The **show profile switch acl** command displays the indicated ACL profile. If no specific profile is selected all created ACL profiles are shown. If an ACL is linked to an IP interface the number of matches for each rule is displayed.

This procedure describes how to display a certain ACL profile.

Mode: Administrator execution or any other mode, except the operator execution mode

Step	Command	Purpose
1	device#show profile acl name	Displays the ACL profile <i>name</i> .

Example: Displaying an ACL entry

The following example shows how to display the ACL profile named WAN_RX.

```
device#show profile acl WAN_RX
```

Examples

Denying a Specific Subnet

Figure 66 shows an example in which a server attached to network 172.16.1.0 shall not be accessible from outside networks connected to IP interface *wan*. To prevent access, an incoming filter rule named *JAMMING* is defined, which blocks any IP traffic from network 172.16.2.0 and has to be bound to IP interface *wan*.

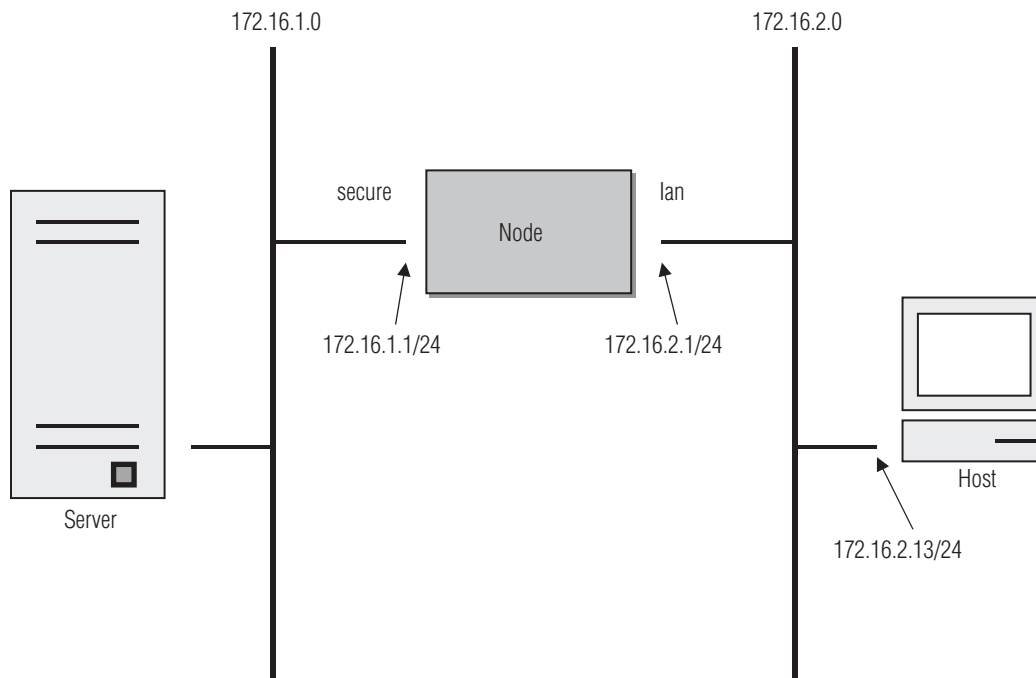


Figure 66. Deny a specific subnet on an interface

The commands that have to be entered are listed below.

```
172.16.2.1>enable
172.16.2.1#configure
172.16.2.1(cfg)#profile acl JAMMING
172.16.2.1(pf-acl)[JAMMING]#deny ip src-ip 172.16.2.0/24 dest-ip 172.16.1.0/24
172.16.2.1(pf-acl)[JAMMING]#permit
172.16.2.1(pf-acl)[JAMMING]#exit
172.16.2.1(cfg)#context ip ROUTER
172.16.2.1(cfg-ip)[ROUTER]#interface wan
172.16.2.1(if-ip)[lan]#use profile acl in JAMMING
172.16.2.1(if-ip)[lan]#exit
172.16.2.1(cfg-ip)#copy running-config startup-config
```

Denying Traffic Between Two Interfaces

Figure 67 shows an example in which there are two LANs, neither of which should be accessible from the other, but both of which should have access to the WAN.

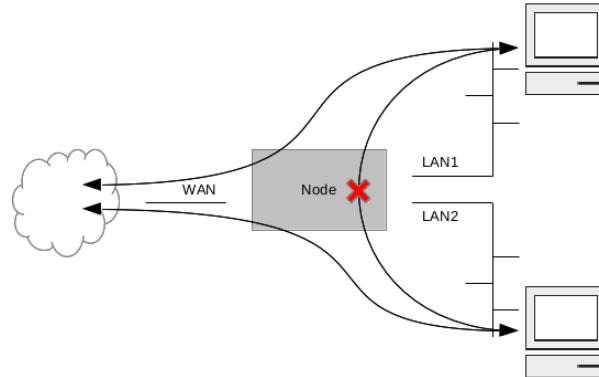


Figure 67. Deny traffic between two interfaces

The commands that have to be entered are listed below.

```
device>enable
device#configure
device(cfg)#profile acl DENY
device(pf-acl)[DENY]#deny
device(pf-acl)[DENY]#exit
device(cfg)#context ip ROUTER
device(cfg-ip)[ROUTER]#interface LAN1
device(cfg-ip)[LAN1]#use profile acl in DENY to interface LAN2
device(cfg-ip)[LAN1]#exit
device(cfg-ip)[ROUTER]#interface LAN2
device(cfg-ip)[LAN2]#use profile acl in DENY to interface LAN1
device(cfg-ip)[LAN2]#exit
device(cfg-ip)[ROUTER]#copy running-config startup-config
```

Permit Only Traffic Generated From LAN

In this example, clients on the LAN are able to connect to FTP (port 21), SSH (port 22), and HTTP (port 80) servers on the WAN as well as to ping them, but any traffic received from the WAN that was not initiated by a client on the LAN is denied.

```
device>enable
device#configure
device(cfg)#profile acl PERMITTED_APPLICATIONS
device(pf-acl)[PERMITTED_APPLICATIONS]#permit protocol tcp dest-port 20,22,80
device(pf-acl)[PERMITTED_APPLICATIONS]#permit protocol icmp
device(pf-acl)[PERMITTED_APPLICATIONS]#exit
device(cfg)#profile acl STATEFUL_FIREWALL
device(pf-acl)[STATEFUL_FIREWALL]#permit connection-state established related
device(pf-acl)[STATEFUL_FIREWALL]#exit
device(cfg)#context ip ROUTER
device(cfg-ip)[ROUTER]#interface WAN
device(if-ip)[WAN]#use profile acl in STATEFUL_FIREWALL
device(if-ip)[WAN]#use profile acl out STATEFUL_FIREWALL
device(if-ip)[WAN]#use profile acl out PERMITTED_APPLICATIONS
```

Chapter 39 **Classifier Configuration**

Chapter contents

Introduction	451
About the Classifier	451
What the Classifier Does	451
How the Classifier Works	451
Classifier Configuration Task List	452
Mapping the Goals of the Classifier	452
Creating a Classifier Profile and Enter Configuration Mode	452
Adding a Rule to the Current Classifier Profile	453
Binding and Unbinding a Classifier Profile To/From an IP Interface to Tag Incoming/Outgoing Packets	453
Binding and Unbinding a Classifier Profile to Tag Locally-generated Packets	455
Displaying a Classifier Profile	456

Introduction

This chapter provides an overview of Trinity's packet classifier and describes the tasks involved in its configuration.

About the Classifier

This section briefly describes what the classifier does, as well as why and when to configure the packet classification.

What the Classifier Does

The classifier is part of the overall Quality-of-Service architecture of Trinity. As depicted in figure 63 on page 424 the classifier groups packet flows into virtual traffic-classes. Other network components in Trinity are able to make routing and quality-of-service decisions based on this traffic-class. For example, the traffic-class may be used for the following:

- To perform **policy routing** by consulting special routing tables for certain traffic classes
- To restrict access to the device by rejecting all packets belonging to a traffic-class in an access control list (**ACL**)
- To tag packet headers with class-of-service information when sending the packet to a remote host (see **service-policy profile**)

That is, the classifier makes Trinity devices aware of different services whereas the ACL, policy routing and the service-policy profile decide how to actually treat them.

How the Classifier Works

The classifier consists of an ordered set of rules. Each rule sets the traffic-class of certain packets dependent on their header fields. A rule consists of the following two parts:

- (a) A condition part, which inspects each packet based on over 20 criteria, such as the source/destination address and port, the packet length, the IP TOS field, etc. (see Chapter 40, "[Packet Matching](#)" on page 458 for the criteria available for matching packets).
- (b) An action part, which sets a traffic-class tag to all packets that are matching the condition (a).

The order of the rules in a classifier profile is significant. Each entry is processed in the order it appears in the configuration file. As soon as the condition part (a) matches, the traffic class is set (b) and no further rules are processed.

Classifier rules are grouped together in profiles. This allows you to bind the same profile (the same set of rules) to several IP interfaces, for example. The order in which the profiles are bound is significant. Each profile is processed in the order in which it is bound in the configuration file. If an entry in one of the profiles matches, the remaining bound profiles are not processed any further.

[Figure 68](#) shows that there are three locations in the routing path where the classifier may examine packets and assign them an internal traffic-class. In each of those locations, a list of classifier profiles can be bound. Quite early after receiving a packet over a port or circuit, the input classifier of the bound IP interface has a chance to classify the packet. In addition, after routing a packet to an egress IP interface, the output classifier of that interface may modify the traffic-class of the packet. Finally, each locally-generated packet may be tagged on the local pseudo-interface before it is routed to an egress interface. [Figure 34](#) on page 283 shows the exact order in

which each packet traverses the classifier and other packet-processing services.

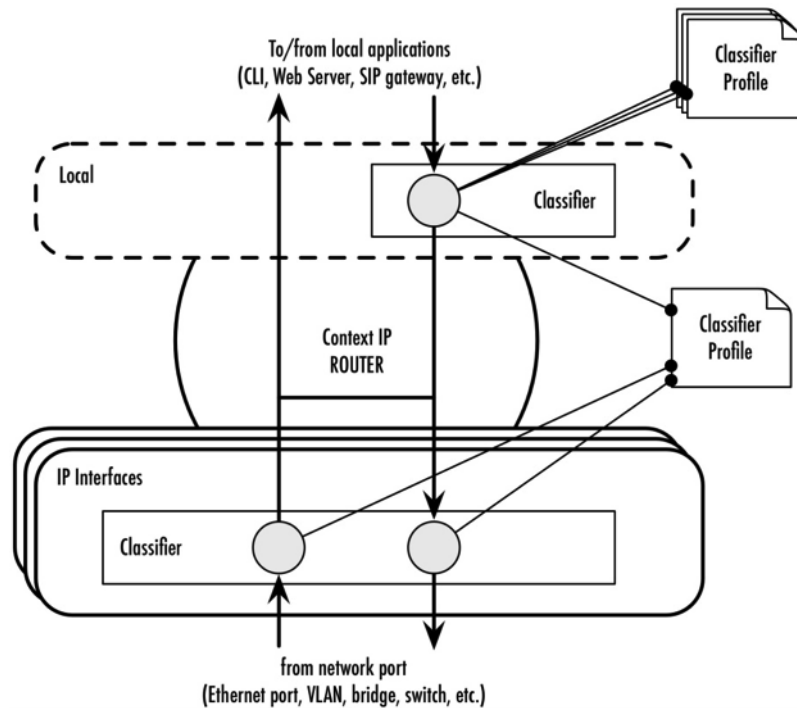


Figure 68. Locations in the routing path where packets may be classified

Classifier Configuration Task List

To configure the classifier, perform the tasks in the following sections.

Mapping the Goals of the Classifier

To create a classifier profile you must:

- Assign a unique name to the classifier profile
- Define packet-filtering criteria
- Define the traffic-class to be set

Before you begin to enter the commands that tag packets with a traffic-class, be sure that you have planned how many traffic-classes are needed, what they mean and what classifier rules are needed to achieve your goals.

Note The access control list (ACL) may rely on traffic-class tags to accept or reject packets. Thus editing a (bound) classifier profile is dangerous as you could potentially lock yourself out forever. Therefore, we recommend editing complex QoS scenarios offline within a configuration file and downloading the configuration file later via TFTP to our Trinity device.

Creating a Classifier Profile and Enter Configuration Mode

This procedure describes how to create a classifier profile and enter the classifier profile configuration mode.

Mode: Administrator exec

Step	Command	Purpose
1	device(cfg)#profile classifier <profile-name>	Creates the classifier profile <profile-name> and enters its configuration mode.

The parameter <profile-name> is the identifier by which the profile will be known. Submitting this command enters into the classifier profile configuration mode, where you can enter the individual classifier rules to set the traffic-class of packets based on filtering conditions.

Use the **no** form of this command to delete a classifier profile. You cannot delete a classifier profile if it is currently bound to an interface or to the *local* pseudo-interface. When you modify a classifier profile, the new settings immediately become active.

Example: Create a classifier profile

In the following example, the classifier profile name *WAN_RX* is created.

```
device>enable
device#configure
device(cfg)#profile classifier WAN_RX
device (pf-cls)[WAN_RX]#
```

Adding a Rule to the Current Classifier Profile

The **match** command is used to define a classifier rule, which matches for some criteria in the packet and sets a traffic-class tag. This procedure describes how to create a classifier rule.

Mode: Profile classifier

Step	Command	Purpose
1	device(pf-cls)[<profile-name>]#match set traffic-class traffic-class	Creates a classifier rule that sets the traffic-class tag to <i>traffic-class</i> if the packet matches all criteria specified in <i>match</i> . See Chapter 40, "Packet Matching" on page 458 for the syntax of the <i>match</i> specifier.

Example: Set the traffic class to *MGMT* for packets (UDP or TCP) sent to destination port 23 (Telnet) or 80 (HTTP).

```
device(cfg)#profile classifier WAN_RX
device(pf-cls)[WAN_RX]#match protocol udp dest-port 23,80 set traffic-class MGMT
device(pf-cls)[WAN_RX]#match protocol tcp dest-port 23,80 set traffic-class MGMT
```

Binding and Unbinding a Classifier Profile To/From an IP Interface to Tag Incoming/Outgoing Packets

To filter incoming or outgoing packets, one or more classifier profiles can be bound to an IP interface. This allows configuring a different classifier for ingress/egress traffic over each IP interface.

The **use** command is used to bind a classifier profile to an IP interface. This procedure describes how to bind a classifier profile to filter incoming/outgoing packets on an IP interface.

Mode: IP interface

Step	Command	Purpose
1	device(if-ip)[ifname]#use profile classifier [in out] <profile-name>	Binds the classifier profile <profile-name> to incoming or outgoing packets on the IP interface <i>ifname</i> .

The command offers the following options:

Parameter	Explanation
<i>ifname</i>	Name of the IP interface to which a classifier profile gets bound.
<profile-name>	The name of a classifier profile that has already been created using the profile classifier command. This argument must be omitted in the no form.
in	Specifies that the classifier applies to packets received over this interface.
out	Specifies that the classifier applies to packets sent over this interface.

The **no** form of the **use** command is used to unbind a classifier profile from an IP interface. When using this form, instead of specifying the name of the profile to unbind, you have to give the numeric index of the respective **use** command as it can be found in the current running-config.

Mode: IP interface

Step	Command	Purpose
1	device(if-ip)[ifname]#no use profile classifier [in out] <i>index</i>	Unbinds the classifier profile at <i>index</i> for incoming or outgoing packets from the IP interface <i>ifname</i> .

Parameter	Explanation
<i>ifname</i>	Name of the IP interface to which a classifier profile gets bound.
<i>index</i>	Numeric index of the bind command (type show running-config current-mode to retrieve the index).
in	Specifies that the classifier applies to packets received over this interface.
out	Specifies that the classifier applies to packets sent over this interface.

Example: Bind and unbind several classifier profiles

Bind two classifier profiles in the specified order to incoming packets on interface *WAN* in the default IP context (*ROUTER*)

```
device(cfg)#context ip ROUTER
device(ctx-ip)[ROUTER]#interface WAN
device(ip-if)[ROUTER.WAN]#use profile classifier in WAN_RX
device(ip-if)[ROUTER.WAN]#use profile classifier in GENERIC
```

Show the current configuration of the IP interface *WAN*:

```
device(ip-if)[ROUTER.WAN]#show running-config current-mode
```

```

ipaddress WAN 20.1.1.1/24
ipaddress DHCP
use profile classifier in 1 WAN_RX
use profile classifier in 2 GENERIC

```

Unbind the *WAN_RX* classifier profile from incoming traffic on interface *WAN*:

```
device(ip-if)[ROUTER.WAN]#no use profile classifier in 1
```

Verify that we have unbound the right profile:

```

device(ip-if)[ROUTER.WAN]#show running-config current-mode
ipaddress WAN 20.1.1.1/24
ipaddress DHCP
use profile classifier in 1 GENERIC

```

Binding and Unbinding a Classifier Profile to Tag Locally-generated Packets

Instead of binding a classifier profile to an IP interface, you are able to bind one or more classifier profiles to the *local* pseudo-interface of the IP context. As shown in [Figure 68](#), these profiles are applied to all packets that are generated locally, independent of the IP interface over which they are being sent.

The **use** command is used to bind a classifier profile to the *local* pseudo-interface. This procedure describes how to bind a classifier profile to filter locally-generated packets

Mode: Context IP

Step	Command	Purpose
1	device(ctx-ip)[ROUTER]#local	Enters the configuration mode of the pseudo-interface for locally-generated traffic.
2	device(local)[ROUTER]#use profile classifier out <profile-name>	Binds the classifier profile <profile-name> to locally-generated packets.

The command offers the following options:

Parameter	Explanation
<profile-name>	The name of the classifier profile that has already been created using the profile classifier command. This argument must be omitted in the no form.
out	Specifies that the classifier applies to locally-generated packets.

The **no** form of the **use** command is used to unbind a classifier profile from the local pseudo-interface. When using this form, instead of specifying the name of the profile to unbind, you have to give the numeric index of the respective **use** command as it can be found in the current running-config.

Mode: Context IP

Step	Command	Purpose
1	device(ctx-ip)[ROUTER]#local	Enters the configuration mode of the pseudo-interface for locally-generated traffic.
2	device(local)[ROUTER]#use profile classifier out <i>index</i>	Unbinds the classifier profile at <i>index</i> for locally-generated packets.

Parameter	Explanation
<i>index</i>	Numeric index of the bind command (type show running-config current-mode to retrieve the index).
out	Specifies that the classifier applies to packets sent over this interface.

Example: Bind and unbind a classifier profile for locally-generated packets.

Bind a classifier profile to locally-generated packets in the default IP context (*ROUTER*)

```
device(cfg)#context ip ROUTER
device(ctx-ip)[ROUTER]#local
device(local)[ROUTER]#use profile classifier out GENERIC
```

Show the current configuration of the *local* pseudo-interface:

```
device(local)[ROUTER]#show running-config current-mode
use profile classifier out 1 GENERIC
```

Unbind the *GENERIC* classifier profile from the *local* pseudo-interface:

```
device(local)[ROUTER]#no profile classifier out 1
```

Displaying a Classifier Profile

The **show profile classifier** command displays the indicated classifier profile. If no specific profile is specified, all installed classifier profiles are shown. The **show** command displays a list of all classifier rules, as well as a list of IP interfaces to which it is bound.

This procedure describes how to display one specific or all classifier profiles

Mode: any

Step	Command	Purpose
1	device>show profile classifier [<i><profile-name></i>]	Displays the rules and bindings of classifier profile <i><profile-name></i> , or, if the <i><profile-name></i> parameter is absent, displays all classifier profiles.

Example: Display a classifier profile

The following example shows how to display the classifier profile named WAN_RX.

```
device>show profile classifier WAN_RX
Classifier Profile: WAN_RX
-----

Rules
-----
match protocol udp dest-port 23,80 set traffic-class MGMT
match protocol tcp dest-port 23,80 set traffic-class MGMT

References: 1
-----
IP Interface(s) (inbound): WAN (ROUTER)
```

Chapter 40 Packet Matching

Chapter contents

Introduction	459
Criteria	459
Connection State	459
Traffic Class	459
Source MAC Address	459
Ethernet Packet Type	459
ToS	459
Precedence	460
DSCP	460
ECN	460
Length	460
TTL	460
Protocol	460
Source IP Address	460
Destination IP Address	460
ICMP Type/Code	460
Source Port	460
Destination Port	460
TCP Flags	460
TCP Option	460
TCP MSS	460
Command Line Syntax	461
Examples	463

Introduction

Several features perform operations on packets based upon certain criteria, such as, their headers. For example, the Classifier feature assigns packets to different traffic classes, the Policy Routing feature routes packets according to different tables and the ACL feature permits or denies packets, depending on the same criteria. All of these features use the same command line syntax to specify which packets match. This chapter lists the criteria that may be used to match packets and specifies the command line syntax.

Criteria

This section lists the criteria that may be used to match packets.

Connection State

This matches packets based on the connection state, which include the following:

- **New**—This matches the first packet the router has received for a given connection. For example, the first packet in an FTP connection is a TCP SYN packet sent from the client to the server. This packet is **new**.
- **Established**—This matches replies to **new** packets, and all subsequent packets in the given connection. For example, when an FTP server responds to a client with a TCP SYN/ACK packet, this packet is **established**.
- **Related**—This matches packets from a connection initiated by another **established** connection. For example, the packets belonging to an FTP-data connection are **related** to the FTP-control connection.
- **Invalid**—This usually indicates a system error, such as being out of memory.

An ACL rule may be added to match packets in **established** and **related** connections in order to implement a stateful firewall.

Traffic Class

This matches packets that have been assigned to a given traffic class by a Classifier rule.

Source MAC Address

This matches packets with a given source MAC address in the Ethernet header.

Note This is only valid for packets received by the router and not for packets generated by the router itself.

Ethernet Packet Type

This matches packets based on whether they are Ethernet unicast packets, broadcast packets (i.e. the destination MAC address is FF:FF:FF:FF:FF:FF), or multicast packets (i.e. the destination MAC address has bit 01:00:00:00:00:00 set).

ToS

This matches packets based on the ToS bits in the IPv4 ToS field.

Note This only applies to IPv4 packets and not IPv6 packets.

Precedence

This matches packets based on the precedence bits in the IPv4 ToS field.

Note This only applies to IPv4 packets and not IPv6 packets.

DSCP

This matches packets based on the DSCP bits in the IP DiffServ field.

ECN

This matches packets based on the ECN bits in the IP DiffServ field.

Length

This matches packets based on the IP payload length.

TTL

This matches packets based on the IPv4 Tim-To-Live or IPv6 Hop Count field.

Protocol

This matches packets based on the IP protocol, for example, ICMP, ICMPv6, TCP, or UDP.

Source IP Address

This matches packets based on the IP source address.

Destination IP Address

This matches packets based on the IP destination address.

ICMP Type/Code

This matches ICMP or ICMPv6 packets based on the ICMP type, and optionally the ICMP code.

Source Port

This matches TCP, UDP, and SCTP packets based on the source port.

Destination Port

This matches TCP, UDP, and SCTP packets based on the destination port.

TCP Flags

This matches TCP packets based on the TCP flags: SYN, ACK, FIN, RST, URG, and PSH.

TCP Option

This matches TCP packets based on whether or not a given TCP option is set.

TCP MSS

This matches TCP packets based on the maximum segment size (MSS).

Note This only applies to TCP SYN and SYN/ACK packets because the MSS is only negotiated during the TCP handshake.

Command Line Syntax

This section explains how to specify the criteria listed above using the CLI.

The following syntax may be used at the CLI to specify the packet match criteria.

Note Multiple criteria may be specified in the same command, in which case, all of the specified criteria must be met for a packet to be considered a match.

Table 27. Command Line Syntax

Criterion	Syntax	Notes
Connection State	connection-state [new] [established] [related] [invalid]	If the connection-state keyword appears, at least one state must be specified.
Traffic Class	traffic-class <name>	The traffic class must have been previously assigned to the packet by the Classifier.
Source MAC Address	src-mac <aa:bb:cc:dd:ee:ff>	
Ethernet Packet Type	packet-type {unicast broadcast multi-cast}	
ToS	tos {<0..15> maximize-cost maximize-reliability maximize-throughput minimize-cost minimize-delay normal} [mask <0..15>]	An optional mask may be specified to limit which bits are examined.
Precedence	precedence <0..7>	
DSCP	dscp {<0..63> af11 ... ef} [mask <0..63>]	An optional mask may be specified to limit which bits are examined.
ECN	ecn <0..3> [mask <0..3>]	An optional mask may be specified to limit which bits are examined.
Length	length <0..65535>[..<0..65535>]	A single length, e.g. 500, or a range, e.g. 28..1480, may be specified. Note that either the minimum or maximum may be omitted in which case it is implied to be 0 or 65535, respectively.
TTL	ttl {eq lt gt} <0..255>	Match packets having a TTL (IPv4) or Hop Count (IPv6) equal to, less than, or greater than the given value.
Protocol	protocol {<0..255> icmp icmpv6 tcp udp}	
Source IP Address	src-ip address	The address may be an IPv4 or IPv6 host, e.g. 192.168.1.1, subnet, e.g. 192.168.1.0/24, or range, e.g. 192.168.1.1-192.168.1.10.
Destination IP Address	dest-ip address	The address may be an IPv4 or IPv6 host, e.g. 192.168.1.1, subnet, e.g. 192.168.1.0/24, or range, e.g. 192.168.1.1-192.168.1.10.
ICMP Type/Code	icmp-type {0..255> echo-reply destination-unreachable ...} [<0..255>]	The first value is the type and the second, optional value is the code. You must also specify <code>protocol icmp</code> to use this criterion.

Table 27. Command Line Syntax

Criterion	Syntax	Notes
ICMPv6 Type/Code	<code>icmpv6-type <0..255> echo-reply destination-unreachable ...> [<0.255>]</code>	The first value is the type and the second, optional value is the code. You must also specify <code>protocol icmpv6</code> to use this criterion. Note ICMPv6 types may have the same name as an ICMP type, but a different numeric value.
Source Port	<code>src-port <0..65535></code>	The port may be a single value, a list, or a range. For example, <code>22,22,23,80,1024..65535</code> , and <code>22,23,80,1024..</code> are all valid. You must also specify <code>protocol tcp</code> , <code>protocol udp</code> or <code>protocol sctp</code> to use this criterion.
Destination Port	<code>dest-port <0..65535></code>	The port may be a single value, a list, or a range. For example, <code>22,22,23,80,1024..65535</code> , and <code>22,23,80,1024..</code> are all valid. You must also specify <code>protocol tcp</code> , <code>protocol udp</code> or <code>protocol sctp</code> to use this criterion.
TCP Flags	<code>tcp-flags [syn {set clear}] [ack {set clear}] [fin {set clear}] [rst {set clear}] [urg {set clear}] [psh {set clear}]</code>	Specify which TCP flags to examine and their value. If the flag is not specified, then it may have either value, set (1) or clear (0). You must also specify <code>protocol tcp</code> to use this criterion.
TCP Option	<code>tcp-option <0..255></code>	You must also specify <code>protocol tcp</code> to use this criterion.
TCP MSS	<code>tcp-mss <0..65535>[..<0..65535>]</code>	A single MSS, e.g. 500, or a range, e.g. 28..1480, may be specified. Note Either the minimum or maximum may be omitted in which case it is implied to be 0 or 65535, respectively. You must also specify <code>protocol tcp</code> to use this criterion.

Examples

```
profile classifier CLASSIFIER_LAN
# Assign packets destined for an SSH, Telnet, or HTTP server to traffic class
# MGMT.
match protocol tcp dest-port 22,23,80 set traffic-class MGMT
# Assign packets destined for an FTP or TFTP server to traffic-class DATA.
match protocol tcp dest-port 20,21 set traffic-class DATA
match protocol udp dest-port 69 set traffic-class DATA

profile acl ACL_WAN
# Allow established and related connections.
permit connection-state established related

context ip

interface LAN
  ipaddress 192.168.1.1/24
  use profile classifier in 1 CLASSIFIER_LAN
  # Route packets from 192.168.1.254 to the 172.16.2.0/24 subnet according to
  # TABLE1.
  route src-ip 192.168.1.254 dest-ip 172.16.2.0/24 dest-table TABLE1
  # Route all other packets according to the DEFAULT routing table.
  route dest-table DEFAULT

interface WAN
  ipaddress 172.16.1.10/24
  use profile acl in 1 ACL_WAN

route DEFAULT
  route default gateway 172.16.1.1

route TABLE1
  route default gateway 172.16.1.2
```

Chapter 41 **SIP Profile Configuration**

Chapter contents

Introduction.....	465
SIP Profile Configuration Task List.....	465
Entering the Configuration Mode for a SIP Profile	465
Mapping from a SIP Disconnect Cause	465
Mapping to a SIP Cause	466
Mapping from a SIP Redirection Reason	466
Mapping to a SIP Redirection Code	466
SIP-Tunneling	466
Autonomous Transitioning for SIP	469

Introduction

The SIP profile specifies disconnect cause mappings from SIP codes to Q.931 causes, and vice versa. As for all profiles, there is a *default* profile at system startup that can be modified. Only those causes that differ from the default mapping have to be configured. If a new profile is created, all mappings are set to their default and are only overwritten if configured. The default mapping in both directions is according to RFC3398 - ISUP to SIP Mapping.

A SIP profile can either be attached to SIP interfaces or to identities. To see how to configure a SIP profile for an interface, see chapter 52, “SIP Interface Configuration” on page 647. For information about SIP profile configuration for identities, see chapter 48, “Location Service” on page 540.

SIP Profile Configuration Task List

This section describes the configuration tasks for SIP profile listed below.

- Enter the configuration mode for a SIP profile (see page 465)
- Map *from* a SIP disconnect cause *to* a Q.931 cause (see page 465)
- Map *to* a SIP cause *from* a Q.931 disconnect cause (see page 466)
- Map *from* a SIP redirection code *to* a Q.931 redirect reason (see page 466)
- Map *to* a SIP redirection code *from* a Q.931 redirect reason (see page 466)

Entering the Configuration Mode for a SIP Profile

The **profile sip** command enters the configuration mode of an existing profile or creates a new one with a specified name. It also destroys an existing profile except the default, which always exists.

Mode: Configure

Step	Command	Purpose
1	[name](cfg)#[no] profile sip<name>	Creates/Destroys a SIP profile or enter the configuration mode of an existing one.

Mapping from a SIP Disconnect Cause

The **map cause from-sip** command maps a specific SIP disconnect cause to a Q.931 cause used by the call control. All causes are pre-defined in the system and are provided by the command.

Mode: Profile SIP

Step	Command	Purpose
1	[name](pf-sip)[name]#map cause from-sip sip-cause to q931-cause	Maps a specific SIP disconnect cause to a Q.931 cause.

Mapping to a SIP Cause

The **map cause to-sip** command can be used to map a call control Q.931 cause to a SIP cause. All causes are pre-defined in the system and are provided by the command.

Mode: Profile SIP

Step	Command	Purpose
1	[name](pf-sip)[name]#map cause to-sip <i>q931-cause to sip-cause</i>	Maps a specific Q.931 disconnect cause to a SIP cause code.

Mapping from a SIP Redirection Reason

The **map redir-reason from-sip** command can be used to map a specific SIP redirect code to a Q.931 redirect reason used by the call control. All redirect codes and reasons are pre-defined in the system and are provided by the command.

Mode: Profile SIP

Step	Command	Purpose
1	[name](pf-sip)[name]#map redir-reason from-sip <i>code to reason</i>	Maps a SIP redirection code to a Q.931 redirection reason.

Mapping to a SIP Redirection Code

The **map redir-reason to-sip** command can be used to map a Q.931 redirect reason to a specific SIP redirect code. All redirect codes and reasons are pre-defined in the system and are provided by the command.

Mode: Profile SIP

Step	Command	Purpose
1	[name](pf-sip)[name]#map redir-reason to-sip <i>reason to code</i>	Maps a Q.931 redirect reason to a SIP redirect code.

SIP-Tunneling

The newly introduced profile sip-tunneling allows tunneling of SIP headers from SIP to SIP calls. Each profile defines a set of SIP headers and a set of SIP messages where the tunneling should be active. A profile bound in incoming direction to a sip interface parses the according messages for the specified headers and forwards the headers to the peer session. A profile bound in outgoing direction inserts all the headers which it get passed from its peer into all specified messages.

Overwriting Headers incoming from new messages overwrite the previously stored headers of the same type. When there are no headers of a certain type in a message then the previously stored headers from older messages still remain active.

Messages triggering Changes of headers which are tunneled do not trigger the sending of new messages. Messages containing the new headers are only sent when the SIP call-flow is in the according state and a signaling message has to be sent anyway.

Excluded headers There is a list of headers which cannot be tunneled because it would destroy the call-flow of the sip calls. These headers are:

- Authorization
- Call-ID
- Contact
- Content-Description
- Content-Disposition
- Content-Encoding
- Content-ID
- Content-Language
- Content-Length
- Content-Transfer-Encoding
- Content-Type
- CSeq
- From
- Max-Forwards
- MIME-Version
- Proxy-Authenticate
- Proxy-Authorization
- RAck
- Record-Route
- Require
- Retry-After
- Route
- RSeq
- To
- Via
- WWW-Authenticate

Caution for these headers These headers may or may not be inserted into sip messages from the normal SIP application. This highly depends on the configured features and the call-flows which happen. When tunneling these headers this may lead to overwriting or duplication of headers. But the device should still be able to perform normal SIP calls, even when having conflict in these headers.

- Accept

- Allow
- Diversion
- Event
- Expires
- History-Info
- Min-Expires
- Min-SE
- P-Asserted-Identity
- P-Preferred-Identity
- Privacy
- Proxy-Require
- Refer-To
- Referred-By
- Replaces
- Server
- Session-Expires
- SIP-If-Match
- Subscription-State
- Supported
- Timestamp
- Unsupported
- User-Agent

Mode: configure

Step	Command	Purpose
1	node(cfg)#profile sip-tunneling <profile name>	Create a new or enter an existing profile sip tunneling.

Mode: sip-tunneling

Step	Command	Purpose
2	node(pf-tun)[<name>]#[no] header <header name>	Adds or removes a sip header type to be tunneled to or from the profile. The header names are case insensitive.

Mode: sip-tunneling

Step	Command	Purpose
3	<code>node(pf-tun)[<name>]#message { INVITE INFO CANCEL BYE 180 183 200 3xx 4xx 5xx }</code>	Sets the exact list of messages where the profile should be active parsing headers inbound or sending headers outbound.
3	<code>node(pf-tun)[<name>]#message add { INVITE INFO CANCEL BYE 180 183 200 3xx 4xx 5xx }</code>	Adds messages to the list of already configured ones.
3	<code>node(pf-tun)[<name>]#message default</code>	Sets the list of active messages back to the defaults. Which is INVITE, CANCEL, BYE, 180, 183, 200, 3xx, 4xx and 5xx.
3	<code>node(pf-tun)[<name>]#message all</code>	Activates the profile on all possible messages. Which is INVITE, INFO, CANCEL, BYE, 180, 183, 200, 3xx, 4xx and 5xx.
3	<code>node(pf-tun)[<name>]#no message { INVITE INFO CANCEL BYE 180 183 200 3xx 4xx 5xx }</code>	Removes the specified messages from the profile, to not be active on these anymore.

Mode: context cs/interface sip

Step	Command	Purpose
4	<code>node(if-sip)[<name>]#[no] use profile sip-tunneling <profile name> { in out both }</code>	Activates a profile to be used on the sip interface in the specified direction.

Autonomous Transitioning for SIP

The autonomous transitioning command provides simultaneous media for audio and media for image in an INVITE request. The opening of a port for audio and a second port for image allows to switch from a normal call to fax relay T38. This is accomplished by sending packets to the other port, without the need of sending a re-INVITE when fax is detected.

Note This feature should not be enabled when the call is passing through NAT or Firewall.

Mode: Profile SIP

Step	Command	Purpose
1	<code>[name](pf-sip)[name]# [no] autonomous-transitioning</code>	Enables or disables the autonomous transitioning.

Chapter 42 **VoIP Profile Configuration**

Chapter contents

Introduction	471
VoIP Profile Configuration Task List	472
Creating a VoIP Profile	472
Configure Codecs	473
Configuring the Transparent-clearmode codec	475
Configuring the Cisco Versions of the G.726 Codecs	475
Configuring the AAL2-G.726-32k Codec	476
SDPptime Attribute	476
Configuring DTMF Relay	476
Configuring RTP Payload Types	477
Configuring RTP Payload Type for Transparent	478
Configuring RTP Payload Type for Transparent-cisco	478
Configuring RTP Payload Type for Transparent-clearmode	478
Configuring RTP Payload Types for the g726-32k and g726-32k-cisco Coders	478
Configuring RTP Payload Type for Cisco NSE	479
Configuring Cisco NSE for Fax	479
Configuring the Dejitter Buffer (advanced)	479
Enabling/Disabling Filters (advanced)	481
Configuring Fax Transmission	482
T.38 CED Retransmission	485
T.38 No-Signal Retransmission	485
Fax Bypass Method	485
Configuring Fax Failover	486
Configuring Modem Transmission	486
Modem Bypass Method	487
Configuring Packet Side Modem/Fax Answer Tone Detection	487
Disabling Fax/Modem Detection for Voice Calls	487
Media Processing	488
Configuring IP-IP Codec Negotiation	488
Configuring the Preferred Codec for Responses	488
RTP Statistics	489
Examples	489
Home Office in an Enterprise Network	489
Home Office with Fax	491
Soft Phone Client Gateway	493

Introduction

This chapter gives an overview of VoIP profiles, and describes how they are used and the tasks involved in VoIP profile configuration.

A VoIP profile is a container for all datapath-related settings on VoIP connections. The profile settings apply to all calls going through the interface. A VoIP profile can be assigned to VoIP gateways and VoIP interfaces in context CS. If no profile is specified for a particular interface, a profile from the gateway the interface binds to is used instead. [figure 69](#) illustrates the relations between VoIP profiles, gateways and CS interfaces. The following components are configurable:

- Codecs and codec parameters (such as silence suppression, RTP payload type, and audio filters)
- DTMF relay
- Dejitter buffer
- Fax transmission
- Modem transmission

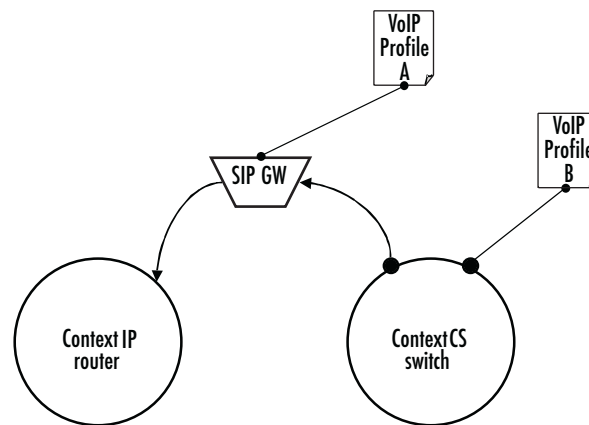


Figure 69. VoIP profile association



Configuring voice datapath options can improve **or degrade** the quality of the transmitted voice data. Many of the default values of these components have configured defaults that should only be changed if required. Misconfiguration can strongly affect the voice quality perceived by the user and the bandwidth requirements of VoIP connections. Be sure you understand the meaning and impact of all commands prior to changing any settings.

VoIP Profile Configuration Task List

The following tasks describe components that can be configured through the VoIP profile:

- Creating a VoIP profile
- Configuring codecs
- SDP Ptime attribute
- Enabling DTMF relay (see page 473)
- Configuring RTP payload types (see page 477)
- Configuring the dejitter buffer (advanced) (see page 477)
- Enabling/disabling filters (advanced) (see page 481)
- Configuring fax transmission (see page 482)
- Configuring modem transmission (see page 486)

If a VoIP profile is modified, the saved modification is applied to all open calls and is valid for all future calls on the gateway or interface using this VoIP profile.

Creating a VoIP Profile

Before configuring voice parameters, a VoIP profile must be created. Each VoIP profile has a name that can be any arbitrary string of not more than 25 characters. When you create the VoIP profile, the VoIP profile configuration mode appears so you can configure VoIP components.

Note The VoIP profile named *default* always exists in the system. It is used by all interface components if there is no other VoIP profile available. If VoIP parameters are the same throughout all interfaces, you can simply change the profile *default* instead of creating a new profile.

Procedure: Create a VoIP profile and enter the VoIP profile configuration mode

Mode: Configure

Step	Command	Purpose
1	device(cfg)#profile voip <i>name</i>	Creates a VoIP profile with name and goes into VoIP profile configuration mode. The newly created profile contains default values for all parameters. If a profile with the same name already exists, only that VoIP profile configuration mode is entered.
2	device(pf-voip)device#...	Configuration steps are described in the following sections.

Example: Creating a VoIP profile

This example shows how to create a VoIP profile named g729_FaxRelay and enter into VoIP profile configuration mode.

```
device>enable
device#configure
```

```
device(cfg)#profile voip g729_FaxRelay
device(pf-voip)[g729_fa~]#...
```

Configure Codecs

The VoIP profile contains a *list* of codecs that forms the set of allowed codecs that can be used to set up a VoIP connection. The list is assembled in order of priority (i.e. the first entered codec is the most preferred one). For each codec in the list, a set of parameters can be configured.



IMPORTANT

Signaling protocols have a codec negotiation mechanism, it is not guaranteed that the first codec in the list is used to set up the connection. Each codec in the list may be used. To make sure that only one codec is possible, configure this codec alone. See how to display the currently configured codecs in a VoIP profile on [page 486](#).



IMPORTANT

The default VoIP profile contains the codecs G.711uLaw and G.711aLaw. If you don't want to use these, you must explicitly remove them from the list.

Procedure: Add a codec to the list (this procedure is valid for all other codecs as well).

Note If you press the **<tab>** key after entering a few letters of a configuration command, the full command name will display or a listing of commands that begin with those letters will display. Press the **<enter>** key to select the desired command.

Mode: Profile VoIP

Step	Command	Purpose
1	device(pf-voip)device#codec g729 tx-length 30 rx-length 30 silence-suppression	<p>Appends codec g729 to the list of codecs. Specifies the payload duration for transmitted RTP packets of this codec, and the maximum supported payload duration for received RTP packets of this codec. Allows silence suppression to be used with this codec.</p> <p>If the codec g729 already existed in the list, its parameters are updated with the entered values. The following codecs are available:</p> <ul style="list-style-type: none"> • g711alaw64k • g711ulaw64k • g723-5k3 • g723-6k3 • g726-16k • g726-16k-cisco^a • g726-24k • g726-24k-cisco • g726-32k • g726-32k-cisco • g726-40k • g727-16k • g727-24k • g727-32k • g729 • netcoder-6k4 • netcoder-9k6 • transparent • transparent-cisco

- a. Cisco does not use the standard ITU G.726 version of G.726, but the ATM AAL2 version. This build series now supports both versions of these codecs. The Cisco G.726 codecs are available in *profile voip* as separate codecs with their name ending in *-cisco*.

Procedure: Remove a codec from the list

Mode: Profile VoIP

Step	Command	Purpose
1	device(pf-voip)device#no codec g729	Remove codec g729 from the list of codecs.

Procedure: Insert a codec at a specific position in the list

Mode: Profile VoIP

Step	Command	Purpose
1	device(pf-voip)device#codec 1 g729 tx-length 30 rx-length 30 silence-suppression	Inserts codec g729 at the first position of the list (most preferred codec). The parameters are the same previously described. If the codec g729 had yet existed in the list, it is moved to the first position of the list, adopting the entered parameter values.

Configuring the Transparent-clearmode codec

To be compatible with RFC4040, transparent-clearmode was made available as a codec in the voip profile. The codec can be used if exclusively packetization and no coding/decoding is needed.

Mode: configure/profile voip

Step	Command	Purpose
1	device(pf-voip)[profile]#[no] codec transparent-clearmode	Allows to use the codec transparent-clearmode.

Configuring the Cisco Versions of the G.726 Codecs

The Cisco versions of codecs are listed in the previous section as separate codecs with their name ending in – Cisco. Trinity supports four Cisco codec versions: *g726-16k-cisco*, *g726-24k-cisco*, *g726-32k-cisco*, and *transparent-cisco*. Three of the codecs are variations of G.726, the fourth is *transparent-cisco*.

The *transparent-cisco* codec provides full compatibility with Cisco's *clear-channel* codec used for transmission of Unrestricted Digital Information over a VoIP (SIP) network.

Cisco does not use the standard ITU G.726 version of G.726, instead it uses the ATM AAL2 version.

All supported Cisco codecs are available in profile voip.

Mode: VoIP *name*

Step	Command	Purpose
1	device(pf-voip)device#codec { g726-16k-cisco g726-24k-cisco g726-32k-cisco ... }	To operate with Cisco's G.726 codecs.

The next table indicates the method of configuring a Cisco-variant codec as the most preferred codec. This example sets the 'transparent-cisco' as number 1, the most preferred.

Mode: VoIP *name*

Step	Command	Purpose
1	device(pf-voip)device#codec 1 transparent-cisco	Configures transparent-cisco as the most preferred codec.

Configuring the AAL2-G.726-32k Codec

It is possible to configure AAL2-G726-32k codec to be available as an option in SDP negotiation.

Mode: profile voip

Step	Command	Purpose
1	device(pf-voip)device#[no] codec [before after] [index] g726-32k-aal2	Enables AAL2-G726-32k codec for negotiation. Please note that specifying this codec automatically disables any other g726-32k codec (g726-32k and g726-32k-cisco). Default: disabled.

It is recommended to specify a custom payload type for the newly introduced AAL2-G726-32k codec (current value is 2).

Mode: profile voip

Step	Command	Purpose
1	device(pf-voip)device#[no] rtp payload-type g726-32k-aal2 <value>	Sets the new payload type value for aal2-g726-32k codec (in range 96..127)

SDPptime Attribute

The SDPptime attribute announces the maximum receive duration for the offered coders. Becauseptime can only be offered on media level and not on a per coder basis, Trinity selects the rx-length of the first configured coder as value for theptime attribute. By default the new attribute is not included in SIP's SDP content. It can be configured to be included with the voip profile's sdpptime-announcement command.

Mode: profile voip

Step	Command	Purpose
1	node(pf-voip)[name]#[no] sdpptime-announcement	Enables/Disables announcement of theptime attribute in SIP's SDP content. Default: disabled

Configuring DTMF Relay

Dual tone multi-frequency (DTMF) tones are usually transported accurately in band when using high bit-rate voice codecs such as G.711. Low bit-rate codecs such as G.729 and G.723.1 are highly optimized for voice patterns and tend to distort DTMF tones. The **dtmf relay** command solves the problem of DTMF distortion by transporting DTMF tones out-of-band or separate from the encoded voice stream as shown in [figure 70](#). SIP uses a mechanism of RTP to reliably transport tones (according to RFC2833).

If "dtmf-relay rtp" (RFC2833) is enabled, the DTMF tones are sent as RTP EVENTS and the DTMF tones are filtered from the RTP media stream. With this enabled, you will only hear a very short tone inband before DTMF detection and filtering kicks in. The default configuration in the voip profile is "dtmf-relay default", which defaults to "dtmf-relay rtp" when SIP is used as the signaling protocol. So, RFC2833 is enabled by default. If you need to send DTMF tones inband you have to disable dtmf-relay, as see in the table below.

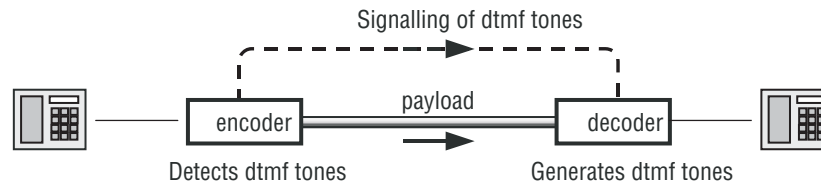


Figure 70. DTMF Relay

This procedure describes how to configure DTMF relay.

Mode: Profile VoIP

Step	Command	Purpose
1	device(pf-voip)device#dtmf-relay signaling [default broadsoft]	The dtmf-relay signaling default command configures the SIP INFO message to be sent in Patton proprietary format. The dtmf-relay signaling broadsoft command configures the SIP INFO message to be sent in Broadsoft proprietary format.
2	device(pf-voip)device#[no] dtmf-relay	Using the no dtmf-relay command, the dtmf are passed INBAND together with the voice in the RTP flow

Step	Command	Purpose
1	device(pf-voip)device#flash-hook-relay [dtmf rtp signaling [broadsoft]]	With the “flash-hook-relay” command the user can chose a different relay method for flash-hook than for the other DTMF keys. The default setting is dtmf.

- **flash-hook-relay dtmf:** flash-hook signals are relayed with the same method as all other DTMF keys. This is the default behavior
- **rtp:** flash-hook signals are sent as rfc2833 RTP events
- **signaling:** flash-hook signals are sent as standard SIP Info messages
- **signaling broadsoft:** flash-hook signals are sent as Broadsoft SIP Info messages

Restrictions:

Since inband-transmission and rfc2833 do not work concurrently, flash-hook-signaling method rtp is not supported when dtmf-relay is disabled or dtmf-relay method is inband. All other combinations work.

Configuring RTP Payload Types

The RTP payload type is one of RTP’s header fields. It identifies the format (e.g. encoding) of the RTP payload and determines the interpretation of the application.

Procedure: Configure RTP NTE payload type

Mode: Profile VoIP

Step	Command	Purpose
1	device(pf-voip)device#rtp payload-type nte <i>payload-type</i>	Specifies the RTP payload-type for named tone events NTE (RFC2833). Default: 101.

Configuring RTP Payload Type for Transparent

The following command configures the RTP payload type used for the transparent codec.

Mode: profile voip

Step	Command	Purpose
1	device(pf-voip)[profile]#[no] rtp payload-type transparent <i><value></i>	Specifies the RTP payload type used for the transparent codec. Value must be between 0 and 127. Default value is 91.

Configuring RTP Payload Type for Transparent-cisco

The following command configures the RTP payload type used for the transparent-cisco codec.

Mode: profile voip

Step	Command	Purpose
1	device(pf-voip)[profile]#[no] rtp payload-type transparent-cisco <i><value></i>	Specifies the RTP payload type used for the transparent-cisco codec. Value must be between 96 and 127. Default value is 116.

Configuring RTP Payload Type for Transparent-clearmode

The following command configures the RTP payload type used for the transparent-clearmode codec.

Mode: profile voip

Step	Command	Purpose
1	device(pf-voip)[profile]#[no] rtp payload-type transparent-clearmode <i><value></i>	Specifies the rtp payload type used for transparent clearmode. Value must be between 96 and 127. Default value is 97.

Configuring RTP Payload Types for the g726-32k and g726-32k-cisco Coders

The following command specifies the RTP payload types for the g726-32k and the g726-32k-cisco coders to be used. It allows changing the payload types to a value in the range of 96 to 127 whereas the default value is 2 for both. Once a payload type has been changed, the 'no' form of the command must be used to go back to the default value.

Mode: profile voip

Step	Command	Purpose
1	device(pv-voip)device#[no] rtp payload-type [g726-32k g726-32k-cisco] <value 96..127>	Defines the RTP payload types for the g726-32k and the g726-32k-cisco coders. Default: 2

Configuring RTP Payload Type for Cisco NSE

Configure the RTP payload-type when transmitting the NSE events. This payload-type is negotiated during call-setup when using SIP.

Named Service Events (NSE) are the Cisco-proprietary version of Named Telephony Events (NTEs). NTEs are defined in RFC 2833. Various telephony signaling events use tones, for example, DTMF. NSEs and NTEs communicate these tones (for representing signaling events), not by the presence of tones, but by sending a binary code representing the tone that is recreated at the destination. Cisco's proprietary NSEs use different values to represent tones and events than the NTEs use.

NSEs are normally sent with RTP payload type 100. The RTP packets have the same source and destination IP addresses and UDP ports as the other packets in the media stream, but differ in the RTP payload types so they can be distinguished from the stream's audio packets.

Mode: Profile VoIP

Step	Command	Purpose
1	device(pf-voip)[name]#rtp payload-type nse payload-type	Specifies the RTP payload-type for Named Signaling Events (NSE). Default: 100

Configuring Cisco NSE for Fax

This command specifies the method to be used for signaling the remote device the RTP Stream has switched to a voice-band Modem transmission. This feature is only available on the SIP protocol. If the command option 'v150-vbd' is selected, a re-invite will be sent even if the current voice coder is configured the same as the modem bypass coder. Furthermore the re-invite contains a gpmd-attribute line with the value 'vbd=yes;ecan=off' in the media description part. This attribute signals the remote device of the new media transmission. If the command option 'default' is selected, the system behavior is the same as before.

Trinity also supports the Cisco NSE standard which uses RFC2833 events for modem transmission over a VoIP (SIP) network. Upon detecting a modem transmission, the called peer issues NSE Event 192. NSE Event 192 indicates a Voice Band Data stream that forces the calling peer to deactivate Voice Activity Detection and to reconfigure the de-jitter buffer for data reception. Afterwards it issues the NSE Event 193 to trigger the calling peer to switch off Echo-Cancellation.

Configuring the Dejitter Buffer (advanced)

Packet networks always introduce a certain amount of jitter in the arrival of voice packets. To compensate for the fluctuating network conditions, a dejitter buffer is integrated in the RTP processing engine. Typical voice sources generate voice packets at a constant rate, the matching voice decompression algorithm also expects incoming voice packets to arrive at a constant rate. However, the packet-by-packet delay inflicted by the network may be different for each packet. As shown in [figure 71](#), the result of the delays is that packets which are sent equally spaced from the left-hand gateway arrive irregularly spaced at the right-hand gateway.

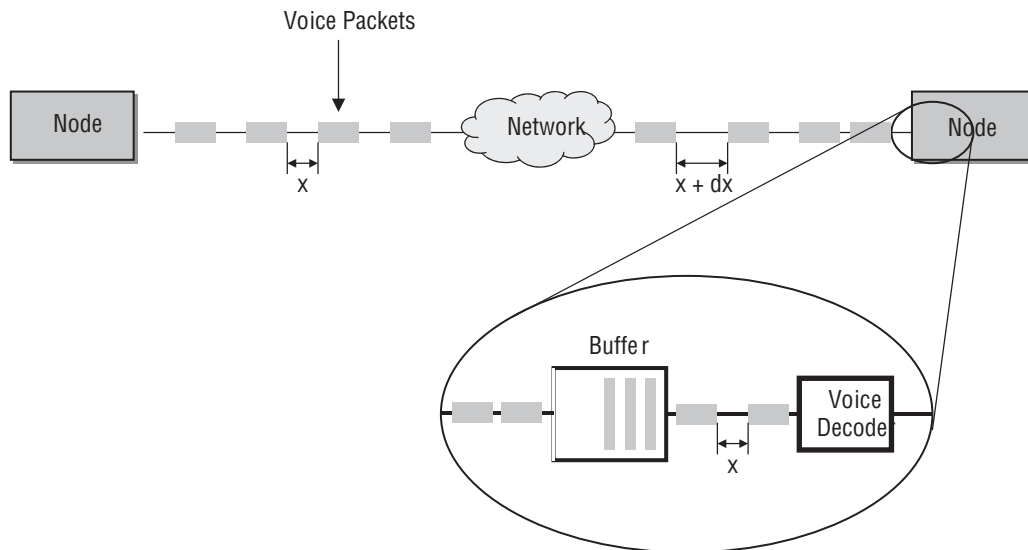


Figure 71. Jitter and de-jitter buffer

The de-jitter buffer delays incoming packets so it can present them to the decompression algorithm at fixed intervals. It will also fix any out-of-order packets by looking at the sequence number in the RTP packets. Such buffering has the effect of smoothing packet flow, and increasing the resiliency of the codec to packet loss, delayed packets, and other transmission effects. The negative side of de-jitter buffering is that it can add significant delay. The de-jitter buffer size is configurable and can be optimized for given network conditions.

The operating modes for the de-jitter buffer are illustrated in [figure 72](#):

- Adaptive—The adaptive buffer automatically adapts to variations in the network's delay characteristics and in general yields the best results for voice conversations.



In the adaptive de-jitter buffer there are parameters that can be configured (such as shrink-speed, grow-step, etc.) that should not be changed unless it is necessary to do so. An incorrect configuration can lead to interoperability problems and loss of service. **Therefore, it is strongly recommended that only experienced users change these parameters.**

- Static—The static buffer is useful for voice conversations if you have specific information about your network's delay characteristics (such as jitter period, etc.), so it should only be used by experienced users.
- Static-data—The static-data mode if you want to create a profile for fax or modem transmission without using the T.38 or fax bypass features described later in this chapter

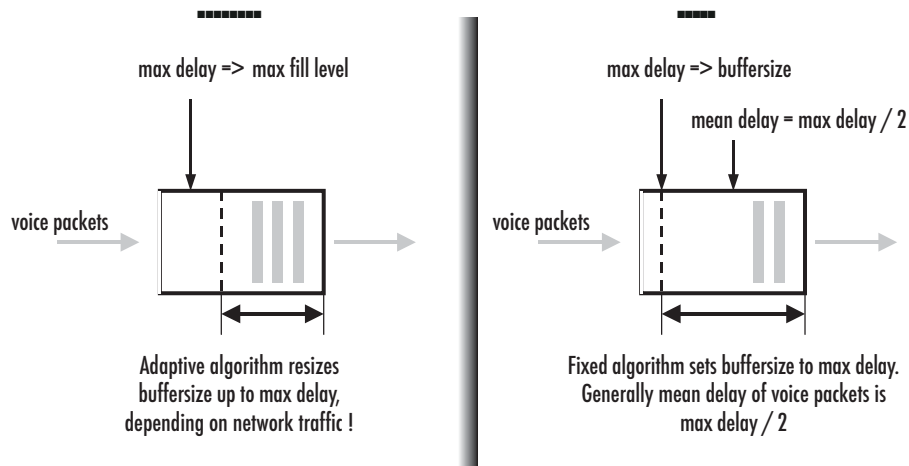


Figure 72. Adaptive versus static de jitter buffer

Procedure: Configure the de jitter buffer.

Mode: Profile VoIP

Step	Command	Purpose
1	<code>device(pf-voip)device#de jitter-mode mode</code>	Specify the de jitter buffer as adaptive, static or static-data.
2	<code>device(pf-voip)device#de jitter-max-delay max-delay</code>	Specify the maximum delay in milliseconds that the de jitter buffer is allowed to introduce. This setting is valid for all modes.

Enabling/Disabling Filters (advanced)

The voice decoder output is normally filtered through a perceptual post-filter to improve voice quality. Likewise a high pass filter is normally used to cancel noises at the coder input. When the communication channels include several Patton devices in tandem as shown in [figure 73](#), sequential post filtering or high pass filtering can cause degrade signal quality. In this case, the user can choose to disable the post-filter and the high-pass-filter.

Note Filtering only occurs with G.723 and G.729 codecs.

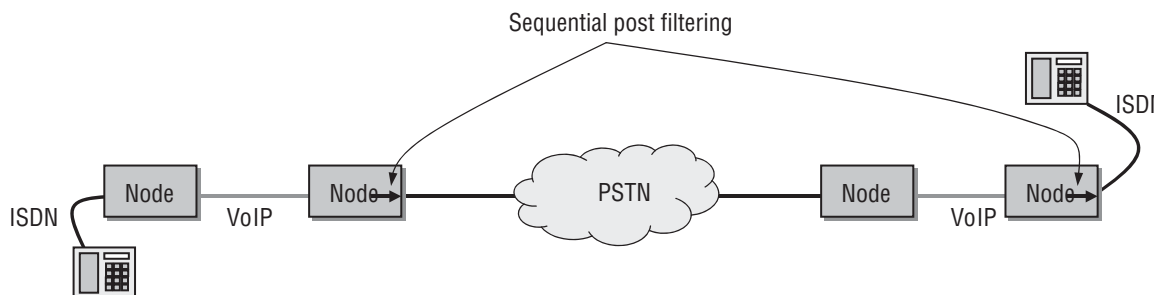


Figure 73. Multiple tandem and sequential post filtering

This procedure describes how to disable post-filtering and high-pass-filtering.

Mode: Profile VoIP

Step	Command	Purpose
1	device(pf-voip)device#no post-filter	Disable decoder output filter
2	device(pf-voip)device#no high-pass-filter	Disable decoder input high pass filter

Example: Disable filters

The following example shows how to disable the decoder output post-filter and the input high-pass filter.

```
device>enable
device#configure
device(cfg)#profile voip myProfile
device(pf-voip)[myProfi~]#no post-filter
device(pf-voip)[myProfi~]#no high-pass-filter
```

Configuring Fax Transmission

Fax is a protocol for electronically transmitting written material in-band over a voice channel. In public switched telephone networks (PSTN), a fax is handled the same way as a voice conversation. A G3 Fax device transforms (modulates) a scanned page into audible tones that are transmitted in-band. The receiving device converts the tones (demodulates) and reconstructs the page. In IP networks, problems can make it difficult to handle a faxed call in the same way as a voice call:

- If one or more RTP packets that transport the voice (tones) are lost, the receiver can't reconstruct what the sender sent.
- Codecs other than G.711 compress the voice streams. They are optimized for compressing voice and not modulated data. Compressing and decompressing always incurs a loss of data.

Trinity provides two solutions for fax transmission problem, fax bypass, and fax relays:

- Fax bypass—When a fax transmission is detected by the Patton device, it automatically switches to a configured fallback codec that does no or little compression. The dejitter buffer is configured with settings optimized for fax transmission.
- Fax relay—Terminates the fax protocol on the Patton device and sends the reference data over a fax protocol (T.38) to the receiver. Fax relay has a smaller bit-error-rate than bypass.
- Fax failover—When using fax transmission in SIP, you can configure the SIP gateway first to try T.38, but if the remote gateway does not support T.38, it will automatically fall back to a high-rate codec.

Both solutions require changing codecs during an established call, which imposes several requirements on the signaling protocol and the remote gateway. Make sure these requirements are met when configuring a fax transmission mode.

Figure 74 illustrates the difference between Fax relay and Fax bypass.

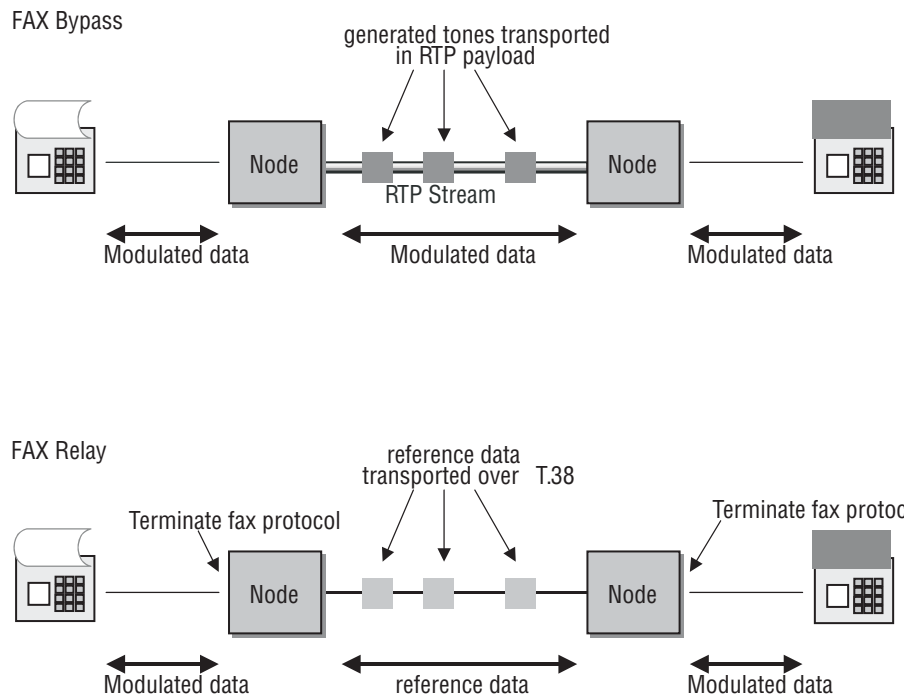


Figure 74. Fax relay and Fax bypass

Fax transmission modes are organized the same way codecs are: there is an ordered list of fax transmission modes; the most preferred fax transmission mode is the first one in the list.

Procedure: Configure fax bypass

Mode: Profile VoIP

Step	Command	Purpose
1	device(pf-voip)device#fax transmission bypass (g711alaw64k g711ulaw64k) [tx-length <ms>] [rx-length <ms>]	Configures the fax bypass transmission mode and sets the packetization time (optional). Default packetization time: 10ms
2 (optional)	device(pf-voip)device#fax dejitter-max-delay buffer-size	Sets the size of the dejitter buffer during fax transmissions. The operating mode of the dejitter buffer is automatically set to fax optimized static-data mode. Patton recommends that you keep the size for fax transmissions higher than that used for voice, since fax is less sensitive to delay than packet loss. The default value is 200ms which should be nominal for almost any transmission network. In exceptional cases it may be necessary to increase this value (maximum 400ms).

Procedure: Configure fax relay (T.38)

Mode: Profile VoIP

Step	Command	Purpose
1	device(pf-voip)device# fax transmission relay t38-udp	Adds fax relay transmission with T.38 protocol over UDP to the list of fax transmission modes.
2 (optional)	device(pf-voip)device#fax redundancy <i>ls low-speed-redundancy hs high-speed-redundancy</i>	Packet loss can be avoided by transmitting the fax data packets several times. This can be configured separately for low speed and the high speed traffic. The default for both parameters is 0 (no redundant transmission). Note that values greater than 0 provide more reliable transmissions, but consume additional bandwidth.
3 (optional)	device(pf-voip)device# fax dejitter-max-delay <i>buffer-size</i>	For proper operation, a dejitter buffer is used on the receiver. The dejitter period can be set to compensate for the jitter imposed by the network. The default value is 200ms which should be nominal for almost any transmission network. Only in exceptional cases it may be necessary to increase this value (maximum 400ms). The dejitter buffer, by default, applies the operation mode 'static-data', i.e. minimizes the packet loss.
4 (optional)	device(pf-voip)device#fax volume <i>volume</i>	Adjusts the volume of the fax signals re-generated on the receiver side. The volume is in dB, in the range -18.5 ... -3.5 (Default: -9.5dB).
5 (optional)	device(pf-voip)device# fax max-bit-rate { 2400 4800 7200 9600 12000 14400 }	Sets maximum allowed bit-rate for fax relay (Default 14400 Bit/sec).
6 (optional)	device(pf-voip)device# fax detection { ced-tone fax-frames }	Selects the method when fax transmissions are detected: By CED tone or by fax frames (Default: ced-tone). It takes longer to detect Fax frames than CED tones, but the risk of misdetection is minimized.
7 (optional)	device(pf-voip)device#no fax error-correction	Disables error correction mode (Default: enabled). If the error correction mode is disabled, the connected fax devices cannot negotiate error correction mode. Connections with error correction mode enabled are more sensitive to packet loss. Disable error correction mode when packet loss is more than 2–3%. Note Error correction mode does not cancel IP packet loss.

Step	Command	Purpose
8 (optional)	device(pf-voip)device#no fax hdlc	Disables HDLC image transfer (Default: enabled). If HDLC mode is enabled, the Patton device removes bit-stuffing, checks CRCs of fax frames arriving from the PSTN and regenerates the CRCs before sending fax frames towards the PSTN. HDLC can only be enabled together with error correction. Disable HDLC when the fax peer does not support this mode.

T.38 CED Retransmission

Even if the user has configured redundant transmission for low-speed and high-speed packets, the T.30 Indicator messages are not included in this process. If the CED message gets lost, the remote device only receives the CED Tone that is sent in-band. But the transmitted in-band tone may be short due to T.38 switchover or too much distortion, such as by using a low bit-rate voice coder like G.723. In this case it is possible that the remote device never starts the initial T.30 procedure, because it has never received the CED tone. For that reason Trinity sends the CED three times in an interval of 100ms with the same sequence number. With this command, you can disable this feature or set the number of retransmissions to a user defined value.

Mode: profile voip *profile-name*

Step	Command	Purpose
1	device (pf-voip)device#[no] fax ced-retransmission <i>number</i>	Specifies the number of CED retransmissions. Default: 2

T.38 No-Signal Retransmission

Some SIP gateways change their port number when switching from audio to T.38. This behavior causes problems if the Patton device is located on the A-Side behind a NAT. Due to T.30 being a unidirectional protocol and the B-Side is normally the initiator of the T.30 handshaking, the Patton device never receives the initial packets of the B-Side because the NAT ports are not yet opened.

To open the NAT ports, Trinity sends T.38 'no-signal' packets when a codec change is detected. By default Patton device sends three such packets. To adjust the number of 'no-signal' packets, use the following configuration command.

Mode: Configure/profile voip

Step	Command	Purpose
1	device (pf-voip)device#fax nosignal-retransmission [1..5]	Sets how many times a T.38 'nosignal' is retransmitted. Default: 3

Fax Bypass Method

This command specifies the method for notifying the remote device that the RTP Stream has switched to a voice-band FAX transmission. This feature is only available on the SIP protocol. If the command option '**v150-vbd**' is selected, a re-invite is sent even if the current voice coder is configured the same as the fax bypass coder. Furthermore the re-invite contains a gpmd-attribute line with the value '**vbd=yes**' in the media description

part. It signals the remote device of the new media transmission. If the command option 'default' is selected, the system behavior is the same as before.

For a fax transmission over a VoIP (SIP) network, the Cisco NSE standard uses events defined by RFC2833. These events are used for the setup of the fax transmission starting between the calling- and called-peer. Upon detecting a fax transmission, the called-peer issues NSE Event 192. NSE Event 192 indicates the data stream is via a voice band, and it forces the calling-peer to do two things—deactivate voice activity detection and reconfigure the de-jitter buffer for data reception. The option 'nse' enables this fax transmission standard.

Mode: profile voip

Step	Command	Purpose
1	device (pf-voip)device#fax bypass-method { default v150-vbd nse x-fax signaling }	Specifies the fax bypass signaling method. Default: <i>default</i>

Configuring Fax Failover

When using fax transmission in SIP, it is possible to configure the SIP gateway to use T.38 and to fall back to a high-rate codec if the remote gateway does not support T.38. This can be configured as follows:

Mode: profile voip <pf-name>

Step	Command	Purpose
1	device (pf-voip)[pf-name]# fax transmission 1 relay t38-udp	Define T.38 UDP as the first fax transmission method to try
2	device (pf-voip)[pf-name]# fax transmission 2 bypass g711alaw64k	Define G.711 A-Law as the second fax transmission method to try, if T.38 is not supported by the remote gateway.

Note The first codec must always be T.38, while the second one must be a high-rate codec such as G.711, which supports fax transmission.

Configuring Modem Transmission

Modem transmission is similar to fax transmission, except that modem data is always transported in bypass mode. This means that an ordered list of bypass codecs can be defined for modem transmission. If no modem transmission codec is configured, no action is taken to change the codec when modem is detected.

Procedure: Configure modem bypass

Mode: Profile VoIP

Step	Command	Purpose
1	device(pf-voip)device#modem transmission bypass (g711alaw64k g711ulaw64k) [tx-length <ms>] [rx-length <ms>]	Configures the modem bypass transmission mode and sets the packetization time (optional). Default packetization time: <i>10ms</i>

Modem Bypass Method

This command specifies the method to be used for signaling the remote device that the RTP Stream has switched to a voice-band Modem transmission. This feature is only available on the SIP protocol. If the command option *v150-vbd* is selected, a re-invite will be sent even if the current voice coder is configured the same as the modem bypass coder. Furthermore the re-invite contains a *gpmd-attribute* line with the value *vbd=yes;ecan=off* in the media description part. This attribute signals the remote device of the new media transmission. If the command option *default* is selected, the system behavior is the same as before.

Trinity also supports the Cisco NSE standard which uses RFC2833 events for modem transmission over a VoIP (SIP) network. Upon detecting a modem transmission, the *called* peer issues NSE Event 192. NSE Event 192 indicates a Voice Band Data stream that forces the *calling* peer to deactivate Voice Activity Detection and to reconfigure the Dejitter Buffer for data reception. Afterwards it issues the NSE Event 193 to trigger the calling peer to switch off Echo-Cancellation.

Step	Command	Purpose
1	device (pf-voip)device#modem bypass- method { default v150-vbd nse x-modem signaling }	Specifies the modem bypass signaling method. Default: <i>default</i>

Configuring Packet Side Modem/Fax Answer Tone Detection

This feature allows the detection of Modem/Fax answer tones on the packet (RTP) side. It allows instantaneous switching from audio mode to voice-band-data mode without executing a re-invitation of the session or sending Named Signaling Events (NSE). This feature allows Trinity to be interoperable with third party gateways or servers that are not able to indicate a media switch by sending V.150 VBD attributes in re-invites nor sending Cisco's NSE events. When enabled, the Patton device starts observing the incoming RTP stream for three seconds upon reaching the connected state. During this observation period, a detected answer tone must be stable for 300 milliseconds. CED-Tone detection is only active when the Patton device is the calling party. If the CED-Tone net side detection is enabled, the user can select from the behaviors described below:

- **default**—Switch from audio mode to voice-band-data mode without executing a re-invitation of the session.
- **re-negotiation**—Issue a re-invite for T.38. This behavior is only valid when fax preferred codec is T.38.
- **fallback**—Switch in to bypass (a bypass coder must be configured). If the bypass coder is different from the audio coder then a re-invite is sent. This behavior is only valid when fax preferred codec is T.38.

Mode: profile voip

Step	Command	Purpose
1	[device](pf-voip)device#[no] ced net-side-detection [re-negotiation fallback]	Enables/Disables Modem/Fax answer tone detection on the network side. Choose between <i>default</i> (no option) or <i>re-negotiation</i> or <i>fallback</i> scenario.

Disabling Fax/Modem Detection for Voice Calls

To prevent wrong fax/modem detection during a voice call, the media detection feature can be disabled. To allow fax and voice calls on the same Patton device, and because a fax call during setup is not distinguishable

from a voice call, media detection is enabled for the specified time period. If after that time no fax or modem is detected, the detection feature will disable to prevent wrong detections that can disturb the call.

Mode: Profile VoIP

Step	Command	Purpose
1	device(pf-voip)device# [no] media detection-timeout <seconds>	Configures the time period that the fax/modem detection is enabled. When disabled or set to 0, the fax/modem detection is enabled during the whole call. Default: Disabled.

Media Processing

In some cases, it may be desired to detect fax or modem calls on the Smartnode. However, for SIP to SIP scenarios, this was not possible when the Smartnode was not actively transcoding. If both call-ends used the same codec, the Smartnode would release the DSP resources to optimize performance but disabling the possibility to detect fax or modem (media) during the call. A new command has been added to make this performance optimization configurable, so that fax and modem can be detected even when not transcoding..

Note TRANSCODING license is required for this feature

Mode: Interface SIP

Step	Command	Purpose
1	device(pf-voip)[<name>]#media-processing auto	Remove DSP resources if not needed (default).
2	device(pf-voip)[<name>]#media-processing forced	Never remove DSP resources, enabling media detection in any SIP-SIP call.

Configuring IP-IP Codec Negotiation

This command is only available on VoIP devices and applies to calls between two IP endpoints (e.g. SIP-SIP). It is in mode “profile voip”.

Step	Command	Purpose
1	device(pf-voip)[default]#[no] codec negotiation	Enables/disables codec negotiation.

Disabled “codec negotiation” honors the codec lists from each call leg independently, formed out of the remote and local capabilities. The DSP is inserted into the RTP path to make sure each side can use its codec. The DSP is transcoding between the codecs of the two RTP streams. Enabled “codec negotiation” will keep the DSP out of the picture for IP-IP calls and tries to negotiate a common codec for both call legs.

Configuring the Preferred Codec for Responses

The normal codec negotiation procedure adopts the preference set through the order that the codecs are listed in the incoming SDP offer. For example, the first codec is the most preferred one followed by the common codecs between peers according to the ordered list of incoming codecs. With the *preferred codec* feature, the Pat-

ton device can be forced to respond with the preferred codec only in the SDP answer. If the preferred codec is contained in the suggested codecs of the incoming SDP offer, it is going to be the only codec that is supported in the current call independently of its priority order in the suggested codecs list.

- Only a codec that has been configured can be selected as preferred codec for the response.
- A codec that has been selected as the preferred codec for the response cannot be removed in the configured codecs list.
- The codec preference feature is taken into account only if the peer on the other side of the call control does not support the IP-IP codec negotiation.

Step	Command	Purpose
1	device(pf-voip)#[no] response-preferred-codec <codec>	Configures the codec to be used in responses to all incoming and supporting SIP calls. Default: disabled.

Example: Choosing a preferred codec for the response

```
profile voip voipName
  response-preferred-codec <codec>
```

RTP Statistics

The following command allows getting call quality information for SIP - SIP calls where no DSP is involved. The DSP is skipped if both SIP parties agree on the same coder.

If the RTP Statistics feature is enabled, the Patton device monitors the RTP/RTCP streams and provides statistics like Packet Count, Byte Count, Packet Loss, Jitter and Round Trip Time to the SIP application. From that information also the MOS values for the two media-streams are calculated.

For SIP calls where a DSP channel is involved, RTP Statistics are always available.

Mode: profile voip <name>

Step	Command	Purpose
1	node(pf-voip)[name]# [no] rtp statistics	Enables recording of RTP statistics for SIP - SIP calls where no DSP is involved. Default: disabled

Examples

Different applications require different VoIP profiles. This section includes a variety of applications and show how the VoIP profile for these applications would be configured.

Home Office in an Enterprise Network

figure 75 is an example of a home office in an enterprise network. The connection bandwidth is 128 kbps and is of very low quality, so the low bit-rate G.723_6k3 codec is used. Likewise, silence suppression is enabled. Because of the low bit-rate codec, DTMF relay is also enabled. As 80 to 100 ms jitter is anticipated, the dejitter buffer is set to *adaptive* with a maximum delay of 100 ms.

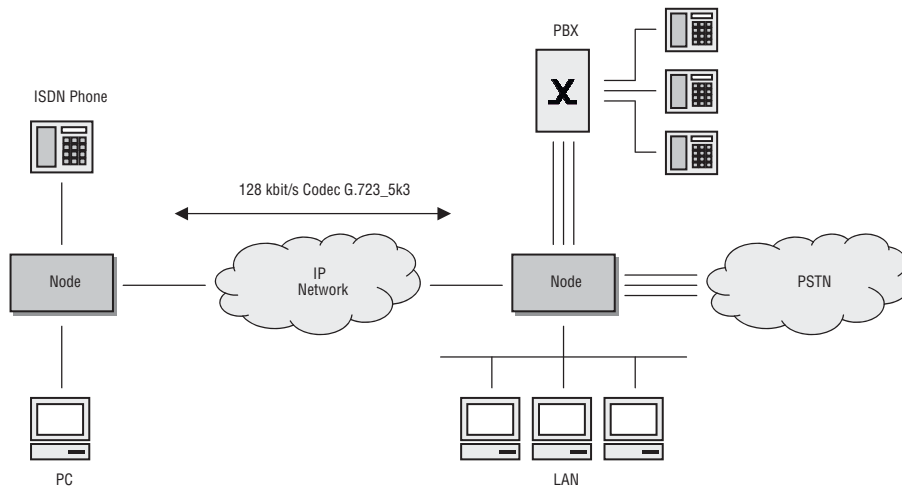


Figure 75. Home office in an enterprise network

First, configure the required CS interfaces (see chapter 45, “[CS Interface Configuration](#)” on page 521) and call routing (see chapter 49, “[Call Router Configuration](#)” on page 561).

Next, configure the voice over IP settings as needed based on the previous description. First we create the VoIP profile with the needed configurations.

```

1 device>enable
2 device#configure
3 device(cfg)#profile voip Wire128kbit
4 device(pf-voip)[Wire128~]#no codec g711aLaw64k
5 device(pf-voip)[Wire128~]#no codec g711uLaw64k
6 device(pf-voip)[Wire128~]#codec g723-6k3 tx-length 30 rx-length 30 silence-sup-
  pression
7 device(pf-voip)[Wire128~]#dejitte-max-delay 100
8 device(pf-voip)[Wire128~]#show profile voip Wire128kbit
VoIP Profile: Wire128kbit
=====

Used:                                     by 0 module(s)

Codecs
-----

G.723 6k3:                                rxlen=30;txlen=30;ss

Fax Transmission
Modem Transmission

Dejitte
-----

Mode:                                   Adaptive
Max. Delay:                                100ms
Max. Packet Loss:                          4/1000
Shrink Speed:                              1

```

```

Grow Step:                1
Grow Attenuation:         1
High Pass Filter:         enabled
Post Filter:              enabled

```

```

Fax
---
```

```

Detection:                CED Tone
T.38 High Speed Redundant Packets: 0
T.38 Low Speed Redundant Packets: 0
Max. Bit Rate:            14400bps
Volume:                   -9.500dB
Error Correction:         enabled
HDLC:                     enabled
Dejitter Max Delay:       200ms

```

```

Modem
-----
```

```

Max. Bit Rate:            14400
Volume:                   -9.500dB
HDLC:                     enabled

```

```

DTMF
----
```

```

Relay:                    enabled
Mute Encoder:             enabled

```

```

RTP
---
```

```

Payload Type NTE:         101

```

Description:

3. Create VoIP profile and give it a name. All settings have default values
- 4., 5. Remove the default codecs G.711alaw and G.711uLaw
6. Add codec g723-6k3 with silence-suppression enabled
7. Allow the dejitter buffer to compensate 100 milliseconds of network jitter.
8. Show the configured profile.

Home Office with Fax

Preconditions are those used in section “Home Office in an Enterprise Network” on page 489: low bandwidth and high jitter. In this example, bandwidth is 256 kbps, what enables us to use the G.729 codec. But since the fax protocol must also be supported, the configuration is extended:

```

1 device>enable
2 device#configure
3 device(cfg)#profile voip g729_FaxRelay
4 device(pf-voip)[g729_Fa~]#no codec g711aLaw64k

```

```

5 device(pf-voip)[g729_Fa~]#no codec g711uLaw64k
6 device(pf-voip)[g729_Fa~]#codec g729 tx-length 20 rx-length 20 silence-suppres-
sion
7 device(pf-voip)[g729_Fa~]#de jitter-max-delay 100
8 device(pf-voip)[g729_Fa~]#fax transmission relay t38-udp
9 device(pf-voip)[g729_Fa~]#fax max-bit-rate 9600
10 device(pf-voip)[g729_Fa]#show profile voip g729_FaxRelay
VoIP Profile: g729_FaxRelay
=====

```

```

Used:                               by 0 module(s)

Codecs
-----

G.729A:                               rxlen=20;txlen=20;ss
T.38 UDP

Fax Transmission
-----

T.38 UDP

Modem Transmission

Dejitter
-----

Mode:                               Adaptive
Max. Delay:                            100ms
Max. Packet Loss:                       4/1000
Shrink Speed:                           1
Grow Step:                              1
Grow Attenuation:                       1
High Pass Filter:                       enabled
Post Filter:                            enabled

Fax
---

Detection:                              CED Tone
T.38 High Speed Redundant Packets:      0
T.38 Low Speed Redundant Packets:       0
Max. Bit Rate:                          9600bps
Volume:                                  -9.500dB
Error Correction:                       enabled
HDLC:                                    enabled
Dejitter Max Delay:                     200ms

Modem
-----

Max. Bit Rate:                          14400
Volume:                                  -9.500dB
HDLC:                                    enabled

```



```

DTMF
----

Relay:                enabled
Mute Encoder:        enabled

RTP
---

Payload Type NTE:    101

```

Description:

3. Create VoIP profile and give it a name. All settings have default values
- 4., 5. Remove the default codecs G.711alaw and G.711uLaw
6. Add codec g729 with silence-suppression enabled
7. Allow the dejitter buffer to compensate 100 milliseconds of network jitter.
8. Enable fax relay over T.38 protocol
9. Limit the maximum bit rate the fax devices can communicate with each other to 9600 kbps
10. Show the configured profile.

Soft Phone Client Gateway

A soft phone client can only use G.711uLaw or G.723 codes, neither of which can use silence suppression, DTMF relay, or fax.

```

1 device>enable
2 device#configure
3 device(cfg)#profile voip softPhone
4 device(pf-voip)[softPho~]#no codec g711aLaw64k
5 device(pf-voip)[softPho~]#codec g723-6k3 tx-length 30 rx-length 30 no-silence-
suppression
6 device(pf-voip)[softPho~]#no dtmf-relay
7 device(pf-voip)[softPho~]#show profile voip softPhone
VoIP Profile: softPhone
=====

Used:                by 0 module(s)

Codecs
-----

G.711 u-law:        rxlen=20;txlen=20
G.723 6k3:         rxlen=30;txlen=30

Fax Transmission
Modem Transmission

Dejitter
-----

```

```
Mode: Adaptive
Max. Delay: 60ms
Max. Packet Loss: 4/1000
Shrink Speed: 1
Grow Step: 1
Grow Attenuation: 1
High Pass Filter: enabled
Post Filter: enabled
```

Fax

```
Detection: CED Tone
T.38 High Speed Redundant Packets: 0
T.38 Low Speed Redundant Packets: 0
Max. Bit Rate: 14400bps
Volume: -9.500dB
Error Correction: enabled
HDLC: enabled
Dejitter Max Delay: 200ms
```

Modem

```
Max. Bit Rate: 14400
Volume: -9.500dB
HDLC: enabled
```

DTMF

```
Relay: disabled
Mute Encoder: disabled
```

RTP

```
Payload Type NTE: 101
```

Chapter 43 **PSTN Profile Configuration**

Chapter contents

Introduction	496
PSTN Profile Configuration Task List	496
Creating a PSTN Profile	496
Configuring the Echo Canceller	497
Configuring Output Gain	497
Configuring Input Gain	498

Introduction

This chapter gives an overview of PSTN profiles, and describes how they are used and the tasks involved in PSTN profile configuration.

A PSTN profile is a container for all datapath-related settings on PSTN connections. It can be assigned to PSTN interfaces in context CS. If no profile is specified in a particular interface, the profile *default* is used. The settings apply to all calls crossing the interface. [figure 76](#) illustrates the relationship between PSTN profiles and CS interfaces. The following components are configurable:

- Echo canceller
- Output gain
- Input gain

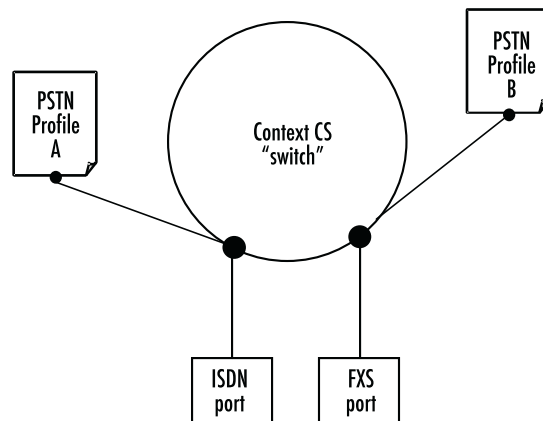


Figure 76. PSTN profile association

PSTN Profile Configuration Task List

The following tasks describe components that can be configured through the PSTN profile.

- Creating a PSTN profile
- Configuring the echo canceler (see page 497)
- Configuring output gain (see page 497)

If a PSTN profile is modified, the saved modification is applied to all open calls and is valid for all future calls on the interface using this PSTN profile.

Creating a PSTN Profile

Each PSTN profile has a name that can be any arbitrary string of not more than 25 characters. When you create the PSTN profile, the PSTN profile configuration mode appears so you can configure PSTN components.

Note The PSTN profile named *default* always exists in the system. It is used by all interface components if there is no other PSTN profile available. If PSTN parameters are the same throughout all interfaces, you can simply change the profile *default* instead of creating a new profile.

Procedure: Create a PSTN Profile and enter the PSTN profile configuration mode.

Mode: Configure

Step	Command	Purpose
1	<code>device(cfg)#profile pstn <name></code> <code> default></code>	Create a PSTN profile with name <i>name</i> and enter PSTN profile configuration mode. The newly created profile contains default values for all parameters. If a profile with name <i>name</i> already exists, only the PSTN profile configuration mode is entered.
2	<code>device(pf-pstn)device#...</code>	Configuration steps as described in the chapters below

Configuring the Echo Canceller

Echoes are reflections of the transmitted signal that result from impedance mismatches in the hybrid (bi-directional 2-wire to 4-wire conversion) device, causing an echo on the wire. Echo cancellation provides near-end echo compensation for this effect as shown in [figure 77](#).



Figure 77. Echo Cancellation

Procedure: Disable echo cancellation.

Mode: Profile PSTN

Step	Command	Purpose
1	<code>device(pf-pstn)device#no echo-canceller</code>	Disable echo canceller (Default: Enabled)

Configuring Output Gain

The output gain determines the voice output volume gain towards PSTN ports as shown in [figure 78](#).

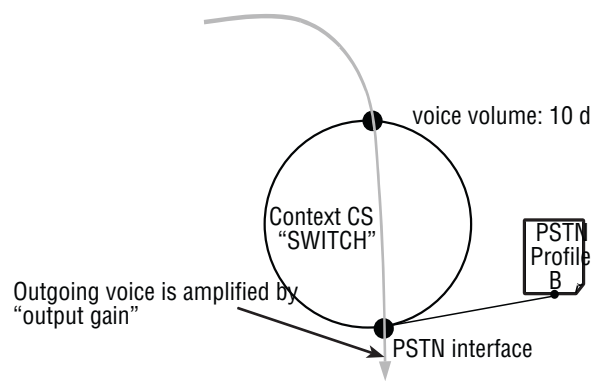


Figure 78. Applying output gain

Procedure: Configure voice output gain.

Mode: Profile PSTN

Step	Command	Purpose
1	<code>device(pf-pstn)device#output-gain gain</code>	Set the output gain to value in dB (Def. 0dB)

Configuring Input Gain

The input gain determines the voice output volume gain from PSTN ports as shown in figure 79.

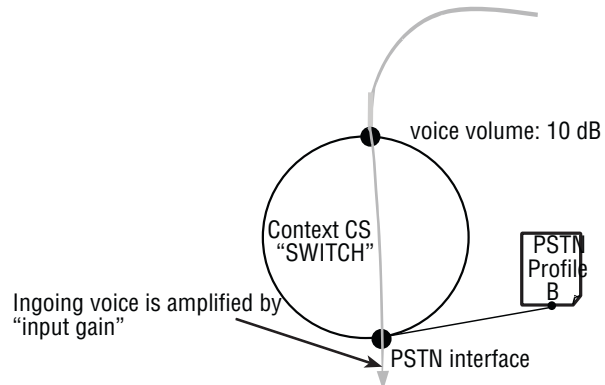


Figure 79. Applying input gain

Procedure: Configure voice input gain.

Mode: Profile PSTN

Step	Command	Purpose
1	<code>device(pf-pstn)device#input-gain gain</code>	Set the input gain to value in dB (Def. 0dB)

Chapter 44 **CS Context Overview**

Chapter contents

Introduction	500
CS Context Configuration Task List	501
Planning the CS Configuration	501
Configuring General CS Settings.....	503
Configuring the clock source	503
Debugging the clock source	504
Selecting PCM law compression	505
Configuring Call Routing.....	505
Creating and Configuring CS Interfaces	506
Specify Call Routing	506
Configuring Dial Tones	506
Configuring Voice Over IP Parameters.....	506
Configuring ISDN Ports	507
Configuring FXS Ports	507
Configuring a SIP VoIP Connection	507
Activating CS Context Configuration.....	508
Planning the CS Context	511
Configuring General CS Settings	512
Configuring Call Routing	512
Configuring VoIP Settings	514
Configuring BRI Ports	514
Configuring an SIP VoIP Connection	515
Activating the CS Context Configuration	515
Showing the Running Configuration	516

Introduction

This chapter gives an overview of the circuit-switching (CS) context and associated components and describes the tasks involved in its configuration. It describes the steps needed to configure voice connectivity and refers to other chapters where a configuration topic is explained in more detail. Before reviewing the content in this chapter, read the configuration concepts as described in Chapter 2, “[Configuration Concepts](#)” on page 57.

The CS context is a high level conceptual entity that is responsible for all aspects of circuit signaling, switching, and emulation. Besides the context switch itself, the CS entity consists of the following (indicated by the shaded area enclosed by a dashed line in [figure 80](#)):

- The CS interfaces
- ISDN
- Tone-set profiles
- Context SIP gateways
- VoIP profiles

The CS Context is enabled by default.

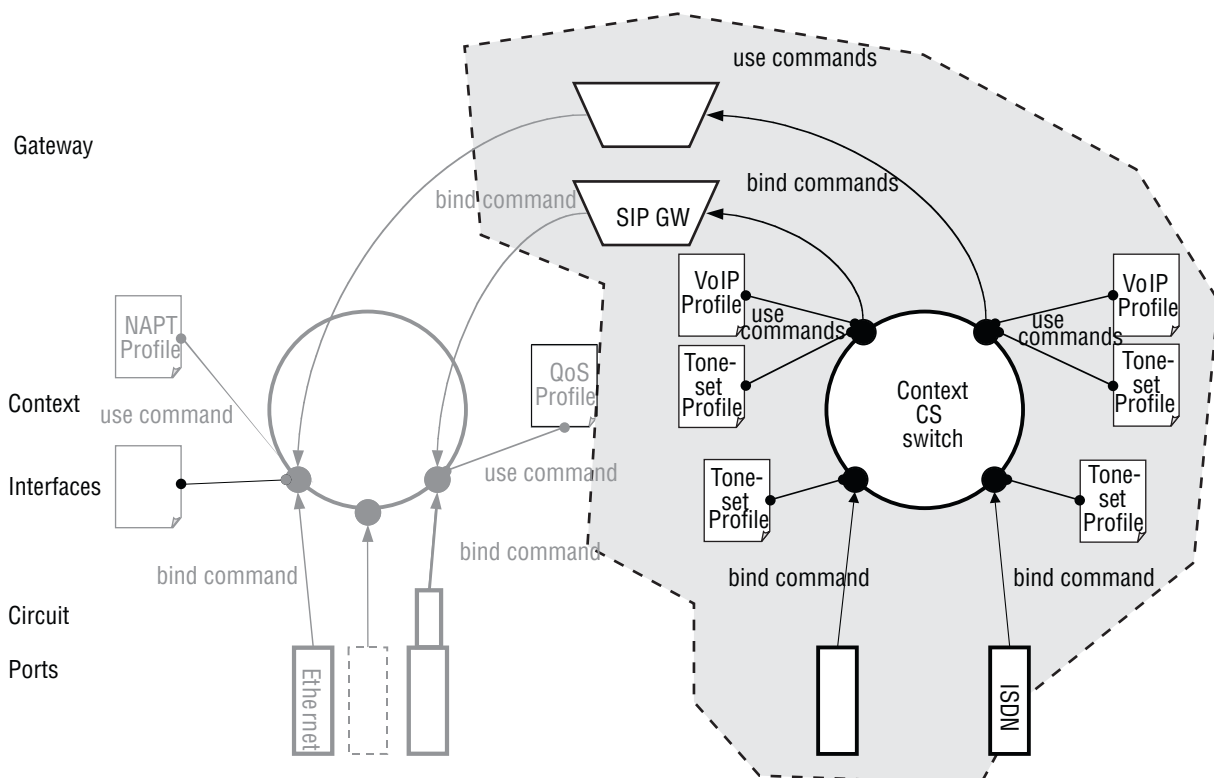


Figure 80. CS context configuration components

The CS context and its associated components route and establish voice calls. For example, the signaling for dial-up circuits is routed and the corresponding voice call circuits are switched between PSTN interfaces and via VoIP interfaces to the Context SIP gateways and the IP context (see section “[Configuring Call Routing](#)” on page 505 for more details).

CS Context Configuration Task List

Information needed for CS entity configuration is distributed among several configuration tasks, depending on its logical content. For example, information pertaining to call routing is described in section “[Configuring Call Routing](#)” on page 505. These configuration tasks can be described in other chapters; thus, to configure call routing you have to refer to Chapter 45, “[CS Interface Configuration](#)” on page 521 and Chapter 49, “[Call Router Configuration](#)” on page 561.

This chapter shows you the relationship between the CS configuration components. We recommend that you perform the CS context configuration in the sequence described below. Many of the parameters have default values that do not need to be changed, which means that you do not have to modify all of the described configuration tasks. In such cases it is stated in the text that you can skip the optional configuration tasks.

1. Planning the CS configuration
2. Configuring general CS settings
3. Configuring call routing
4. Creating and configuring CS Interfaces
5. Configuring dial tones (advanced)
6. Configuring voice over IP settings (advanced)
7. Configuring ISDN ports
8. Configuring a SIP VoIP connection
9. Activating the CS context configuration
10. CS Context Chapter Overview Example

Planning the CS Configuration

There are many policies and factors that can influence the CS context configuration. It depends on what your application is and how your network is configured. Several factors to consider for planning your CS configuration are listed below:

- Application/network scenario
- Peripheral devices, such as PBX or remote VoIP gateway.
- VoIP protocol
- Number and type of physical telephony ports available
- Call routing

Figure 81 Shows a typical application with a remote office in an enterprise network. This example focuses on the Patton device in the remote office. There is an ISDN phone, a personal computer, a connection to the public ISDN network, and a connection to the IP backbone. The VoIP protocol used is SIP with a codec G.711. A call can be routed to the IP backbone and the public ISDN network depending on its prefix and number length.

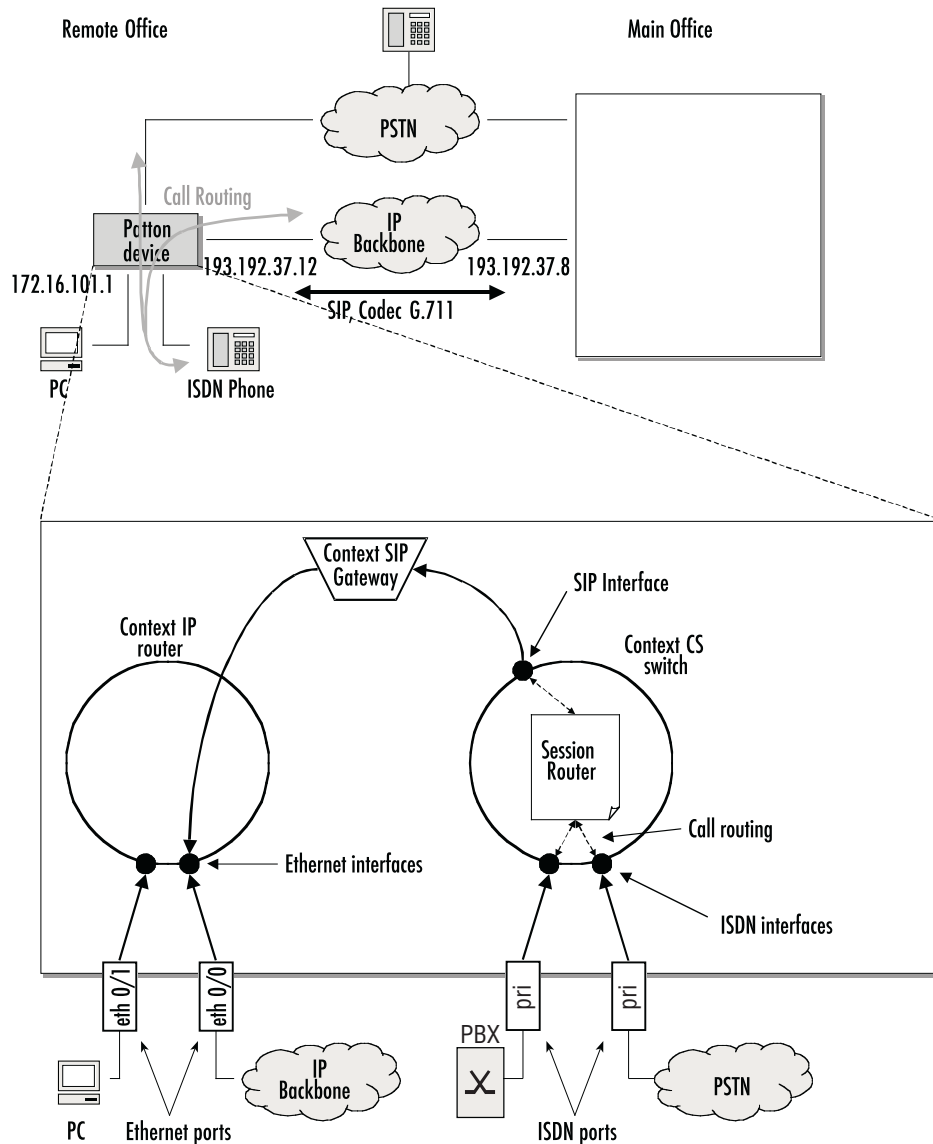


Figure 81. Remote office in an Enterprise network

An application like that shown in figure 81 would require the following CS configuration:

- Since the remote office is connected to the public switched telephone network, the clock-source comes from the corresponding ISDN port. (Described in section “Configuring General CS Settings” on page 503).

Note Be careful when choosing where you get your clock source, if the clock used for packaging the ISDN voice frames is not synchronized with the remote

ISDN clock, bit errors may result (such synchronization problems would probably cause a fax transmission to fail).

- Two PRI ports will be needed, the first port for the ISDN PRI PBX and the second for the public ISDN network (see section “[Configuring ISDN Ports](#)” on page 507).
- Two ISDN interfaces will be needed, each bound to a BRI port (see section “[Configuring Call Routing](#)” on page 505)
- The call router routing tables, and the SIP and ISDN interfaces will have to be configured to support call routing (see section “[Configuring Call Routing](#)” on page 505).

Calls are routed from an ISDN PBX with a number in the range of 1xx–5xx to the main office with a fall-back to the PSTN. All other calls are routed from the ISDN PBX to the PSTN and from the PSTN or main office to the ISDN phone behind the PBX.

- The Context SIP gateway must be configured to use the G.711 codec (see section, “[Configuring Voice Over IP Parameters](#)” on page 506)
- Two Ethernet ports and their corresponding IP interfaces will be needed.

You must not start to configure the CS context and its components until you have finished planning your voice environment. The following chapters explain how to convert the planned voice environment into the Trinity CS configuration. The IP configuration is not a topic in this example. For more information on IP configuration refer to Chapter 21, “[IP Context Overview](#)” on page 280.

Configuring General CS Settings

There are several parameters that cannot be collected into one specific configuration task, because they are independent of the rest of the CS context configuration and apply mostly to an interface card or even to the entire Patton device.

Configuring the clock source

A reference clock is needed for packaging the ISDN voice frames. The reference clock can be generated internally or obtained from an external source (e.g. public ISDN). Patton devices have a feature called ‘Clock Source Hunting’. This feature allows the user to configure an index-based list of clock sources. The source with the lowest index has the highest priority and vice versa. On Patton devices populated with several PRI or BRI ports where more than one port is working in ‘clock slave’ mode, all these ports can be entered in the clock source list. The algorithm behind this feature always takes the first synchronized ‘slave’ port in the list as the current clock source. If the links of all the ports in the list are down or not synchronized, the system is falling back to its internal clock source. It is also possible to enter all PRI or BRI ports of the device into the list, independent of their clock mode. The Clock Source Hunting algorithm ignores all entered ports that are not working in ‘slave’ mode.

Mode: System

Step	Command	Purpose
1	device(sys)#clock-source<index>[1..10]	Enter position of the clock source
2	device(sys)#clock-source after <index>[1..10]	Insert an entry after position 'index' (1-10)
3	device(sys)#clock-source before<index>[1..10]	Adds a BRI port as clock source
4	device(sys)#clock-source bri <slot>[0..]	Adds a E1T1 port as a clock source
5	device(sys)#clock-source e1t1 <slot>[0..]	Move entry at 'index' number of 'positions' up
6	device(sys)#clock-source index up positions	Move entry at 'index' number of 'positions' up
7	device(sys)#clock-source index down positions	Move entry at 'index' number of 'positions' down

Debugging the clock source

To control the system behavior at runtime, there exists a debug clock-source command with the options 'detail' and 'full-detail'.

Mode: Operator execution

Step	Command	Purpose
1	device#[no] debug clock-source (detail 0 full-detail)	Troubleshoots the system clock monitor

Mode: Operator execution

Step	Command	Purpose
1	device#show clock-source	Print system clock information

```
device#show clock-source

Current clock source
=====

internal

Registered clock sources
=====

Name                Capable Sync
e1t1 0 0 0
internal            x      x
```

Selecting PCM law compression

The PCM law-select specifies the voice characteristic compression curve. Two values are possible: *a-Law* (used in Europe) and *μ-Law* (used in the USA).

Procedure: To set the general CS parameters

Step	Command	Purpose
1	<code>[node](cfg)#system pcm law-select { alaw ulaw }</code>	Generates the reference clock internally or specifies a specific port to receive the reference clock. Default: alaw

Configuring Call Routing

Calls through a Patton device can be routed according to a set of routing criteria. The entity that manages call routing is called the *call router*. Calls are routed from one CS interface to another. The call router determines the destination interface for every incoming call. It supports complex call routing and call property manipulation (e.g. number manipulation) functions. See Chapter 49, “[Call Router Configuration](#)” on page 561.

Call routing occurs in the context CS element between several CS interfaces. Accordingly, a CS context and two or more CS interfaces must be created.

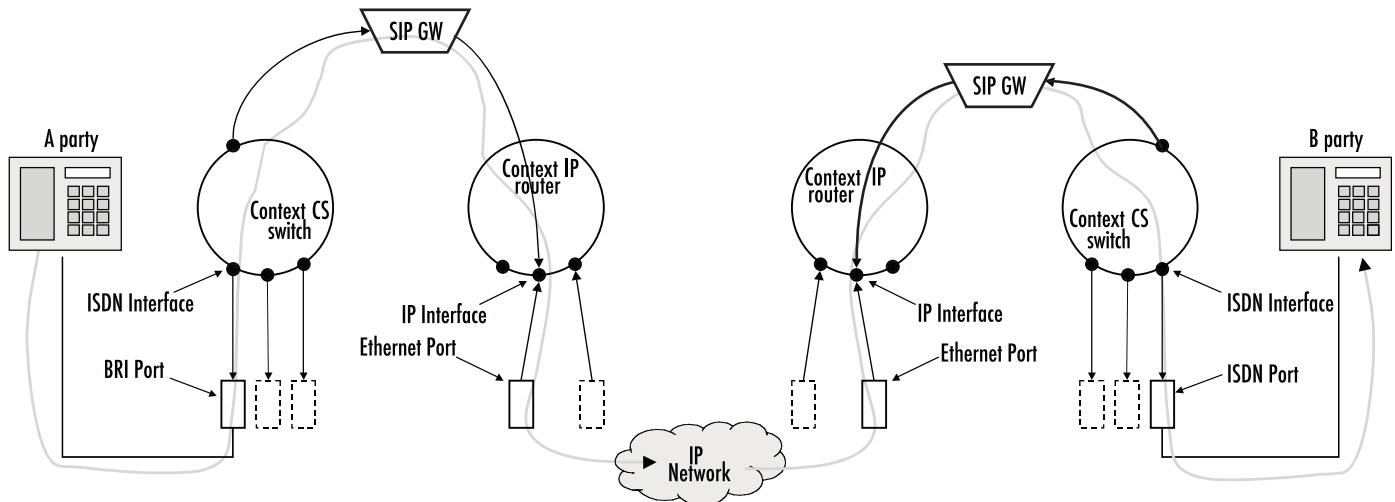


Figure 82. Direct call routing from one Patton device to another

Figure 82 shows a call set up from the A-party on the left to the B-party on the right. The call is routed from the phone on the left-hand side over the ISDN interface directly to an SIP interface. Once it has passed the IP context and the IP network, the other device—from the SIP interface to the ISDN interface and then over the BRI port to the B-party phone—routes the call.

Note Because call routing occurs only in the CS context, in future figures the context IP is omitted. For configuring call routing you have to create the CS interfaces and the call router tables as described in the chapters below. For

simple call routing directly from one interface to another you can even omit router tables.

Creating and Configuring CS Interfaces

Multiple instances of CS contexts are supported. The name of the default instance is *switch*. The name and number of CS interfaces depends on your own configuration. The interfaces on the CS context represent logical connections to other equipment or networks. CS interfaces are used as source and destination in the call router. VoIP CS interfaces are bound to a gateway. Telephony ports are bound to respective interfaces.

Interface names can be any arbitrary string with a maximum of 25 characters. For ease of identification, the interface type can be a part of the name. For examples and information on how to create CS interfaces, refer to Chapter 45, “[CS Interface Configuration](#)” on page 521.

Specify Call Routing

As mentioned previously, for basic call routing you can omit creating call router tables. Trinity offers two levels of call routing:

- Basic interface routing
- Advanced call routing

Basic interface routing allows you to forward all incoming calls on a CS interface directly to a destination CS interface. The call router allows you to route calls to all available CS interfaces, based on a call property such as calling number, destination number and ISDN bearer capability and many more.

We recommend that you first carefully consider what interfaces and call router tables are required to achieve your goals on a sheet of paper, then start creating and configuring CS interfaces, and setting up call router tables.

To configure basic interface routing refer to Chapter 45, “[CS Interface Configuration](#)” on page 521. Other topics that belong to call routing are also explained in this chapter.

To configure advanced call routing in relation to the call router tables refer to Chapter 49, “[Call Router Configuration](#)” on page 561. In this chapter, the differences between basic interface routing and advanced call routing are described in more detail.

Configuring Dial Tones

Trinity supports country-specific, configurable, in-band dial tones that are generated for specific events, for example, alerting, and dialing or busy signals. The tones are configured in tone-set profiles that are used from a specific CS interface.

If no tone-set profile is specified, a default tone-set profile is used. In most cases, the default profile can be used, so you do not need to perform this configuration task.

Configuring Voice Over IP Parameters

In Trinity, there are many configurable parameters that can affect a voice over IP connection.

The *voice over IP* (VoIP) parameters are configured in the VoIP profile. A VoIP profile is used by a SIP interface. All calls going through that interface (see figure 82 on page 505) use the settings in the VoIP profile. The following parameters are configured in the VoIP profile:

- Codecs
- Fax transmission
- Filters
- DTMF relay
- Echo canceller
- Silence compression
- Voice volume
- Dejitter buffer

Refer to Chapter 42, “[VoIP Profile Configuration](#)” on page 470 to configure general VoIP parameters. Some settings can adversely affect the voice quality perceived by the user and the bandwidth requirements of VoIP connections, so be sure you understand the meaning of the commands before changing any settings. Most of the default values of these parameters are adequate, so you generally do not need to perform these configuration tasks.

If no VoIP profile is specified for use on an interface, a default VoIP profile is used. In most cases, the default profile can be used so you just need to change the default VoIP profile.

Configuring ISDN Ports

BRI and E1/T1 ports represent physical ports on the Patton device. The configuration of the ISDN ports depends on the port type (BRI, E1 or T1), and on the connected voice device. To configure the ISDN ports, refer to Chapter 63, “[ISDN Interface Configuration](#)” on page 806.

Configuring FXS Ports

FXS ports represent physical ports on the Patton device. To configure the FXS ports, refer to chapter 68, “[FXS Port Configuration](#)” on page 885.

Configuring a SIP VoIP Connection

To configure a SIP connection, you have to specify the voice codec selection and the call signaling method for the VoIP profile.

When configuring the voice codec for a SIP connection, you must specify the VoIP profile that shall be used on a SIP interface. The VoIP profile contains an ordered list of codecs that shall be used for codec negotiation for all calls routed through this interface. During a call setup, the first codec that is specified in the VoIP profile is taken. For information on how to configure the codecs, refer to Chapter 42, “[VoIP Profile Configuration](#)” on page 470.

You can configure the Context SIP gateway to a registrar with multiple URIs. Optionally, you can configure the Context SIP gateway to send all requests to an outbound proxy or redirect server.

You have several options on how to build a destination URI (To-URI) of an outgoing SIP call. You can use the called party number in conjunction with the specified domain name or you can set a specific URI by the call router, based on other call properties. For examples and information on how to configure the Context SIP gateway, refer to Chapter 60, “[Context SIP Gateway Overview](#)” on page 777. For more on SIP interface configura-

tion, refer to Chapter 52, “[SIP Interface Configuration](#)” on page 647.

Activating CS Context Configuration

After configuring the CS context and its components, the configuration must be activated. This includes binding the physical ports to the virtual CS interfaces and enabling the gateways, ports, and the CS context.

In order to become functional, each interface must be bound from one port where it receives incoming calls and also forwards outgoing calls. Unlike ISDN interfaces, VoIP interfaces must be bound to a Context SIP gateway.

Note The difference between VoIP and PSTN interfaces is that VoIP interfaces are bound to a Context SIP gateway while PSTN ports are bound to a CS interface. After binding, the BRI, E1, or T2 ports must be enabled to become active.

To bind an ISDN port to an ISDN interface, refer to Chapter 63, “[ISDN Interface Configuration](#)” on page 806. To bind an FXS port to an FXS interface, refer to chapter 68, “[FXS Port Configuration](#)” on page 885. Likewise, the Context SIP gateway must be enabled. Additionally, the Context SIP gateway must be bound to a specific IP interface. For more information, refer to Chapter 60, “[Context SIP Gateway Overview](#)” on page 777.

In order to become active, the CS context must be enabled. When recovering from the shutdown status, the CS context and call router configuration is checked and possible errors are indicated. The call router debug monitor can be enabled to show the loading of the CS context and call router configuration. Trinity offers a number of possibilities to monitor and debug the CS context and call router configurations. For example, the call router debug monitor enables you to follow the sequence of tables and functions examined by the call router for each call setup. Refer to Chapter 69, “[Debug and Monitoring](#)” on page 906 for an introduction to the configuration debugging possibilities in Trinity.

Note You can modify the configuration at runtime; changes will be active after 3 seconds. It is not necessary to shut down the CS context before making configuration changes, a newly created or changed configuration is automatically loaded as long as the context CS is not shut down. All active calls are not affected by this reload.

There are several possibilities to show the actual CS context configuration. For more information on the show command, refer to the respective configuration Chapters or to the Chapter 45, “[CS Interface Configuration](#)” on page 521” and Chapter 49, “[Call Router Configuration](#)” on page 561.

Procedure: Show the CS context configuration, enable the call router debug monitor and activate them in the CS context

Mode: Context CS

Step	Command	Purpose
1	device(ctx-cs)[SWITCH]#show call-router config detail <level>	Shows the CS context configuration. <i>Level</i> could be 1..5. Level 1 shows less, level 5 shows all information.
2	device (ctx-cs)[SWITCH]#debug cr detail <level>	Enables the call-router debug monitor. <i>Level</i> could be 1..5. Level 1 only logs errors, level 5 shows all relevant information to track calls through routing tables.
3	device (ctx-cs)[SWITCH]#no shutdown	Enables the CS context, checks the interface and call router configuration
4	device(ctx-cs)[SWITCH]#show call-router status detail <level>	Shows the actual state of the call router. This includes all configured tables as they were read-in from the configuration.

Example: Enable CS Context

The following example shows how to enable the call router debug monitor and how to enable the CS context. It also shows the output from the call router debug monitor.

```

device(cfg)#show call-router config detail 5
Table switch/TAB-ISDN-SERVICE:
Key          Value          Function          Dest-Type          Dest-Name
itc          -              -                dest-interface    IF-LOCAL-BA
-----
unrestricted-digital -          -                dest-interface    IF-LOCAL-BA
default      -              -                dest-table        TAB-DEST-A

Table switch/TAB-DEST-A:
Key          Value          Function          Dest-Type          Dest-Name
called-e164  -              -                dest-interface    IF-LOCAL-BA
-----
0            -              MAP-CAC-ORANGE   dest-interface    IF-LOCAL-BA
00           -              MAP-CLI-MELON    dest-interface    IF-NODE-C
07[4-6]     -              MAP-CAC-APPLE    dest-interface    IF-LOCAL-BA
0336652...  -              -                dest-interface    IF-NODE-B
default      -              -                dest-interface    IF-LOCAL-BA

Table switch/CAC-APPLE:
Key          Value          Function          Dest-Type          Dest-Name
called-e164  called-e164    -                -                  -
-----
(.%)        1055\1        -                -                  -

...

device(cfg)#debug cr
device(cfg)#context cs
device(ctx-cs)[SWITCH]#no shutdown

```

```

02:14:30 CR > Updating tables in 3 seconds...
02:14:33 CR > [SWITCH] Reloading tables now
02:14:33 CR > [SWITCH] Flushing all tables
02:14:33 CR > [SWITCH] Loading table 'TAB-ISDN-SERVICE'
02:14:33 CR > [SWITCH] Loading table 'TAB-DEST-A'
02:14:33 CR > [SWITCH] Loading table 'CAC-APPLE'
02:14:33 CR > [SWITCH] Loading table 'CAC-ORANGE'
02:14:33 CR > [SWITCH] Loading table 'CLI-MELON'
02:14:33 CR > [SWITCH] Loading table 'MAP-CAC-APPLE'
02:14:33 CR > [SWITCH] Loading table 'MAP-CAC-ORANGE'
02:14:33 CR > [SWITCH] Loading table 'MAP-CLI-MELON'
02:14:33 CR > [SWITCH] Loading table 'IF-LOCAL-BA-precallservice'
02:14:33 CR > [SWITCH] Loading table 'IF-PBX-A-precallservice'
02:14:33 CR > [SWITCH] Loading table 'IF-device-B-precallservice'
02:14:33 CR > [SWITCH] Loading table 'IF-device-C-precallservice'
device(ctx-cs)[SWITCH]#

```

Example: Configure the Patton device in an Enterprise Network

Situation: Figure 83 shows an enterprise network with a Patton device configured with a BRI port. A PBX, a LAN, the PSTN, and the company network are connected. The VoIP protocol used is SIP. The voice codec used is G.723, so the DTMF relay is enabled. Because no special dial tones have to be specified, the default tone-set profile is used.

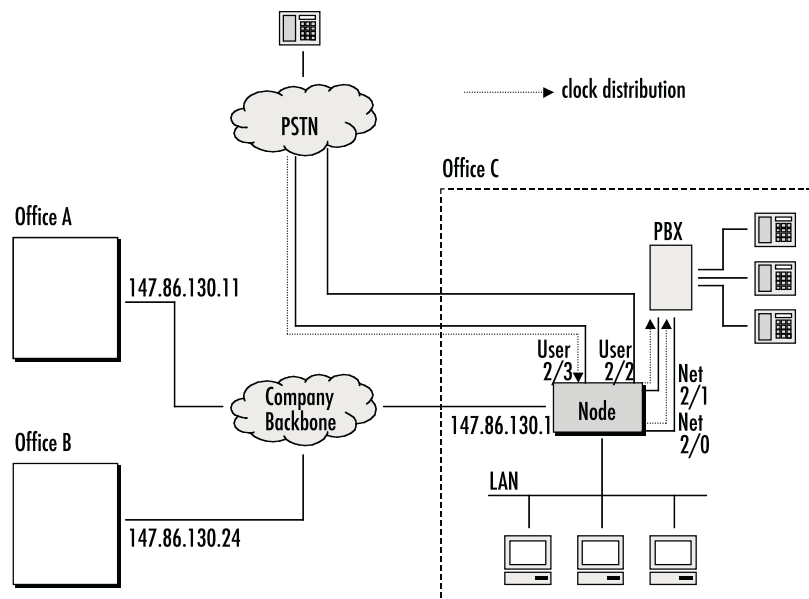


Figure 83. Patton device in an Enterprise network

Call routing is specified as follows:

- Calls from *office C* with number *1xx* to *office A* with a fallback to PSTN
- Calls from *office C* with number *2xx* to *office B* with a fallback to PSTN
- All other calls from *office C* to PSTN

- Calls from *office A* or *B* with number *5xx* to *office C*
- All other calls from *office A* or *B* to the PSTN (local breakout)

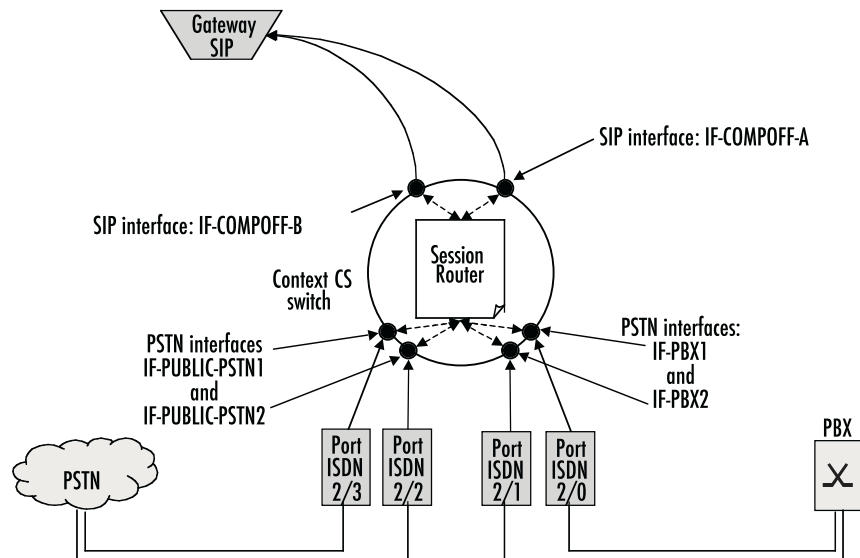


Figure 84. CS Configuration

Planning the CS Context

Based on the criteria used in the previous example, the following configuration information applies (see figure 84):

- It is very important to specify from where to get the clock source for the packaging of the ISDN voice frames. In the example we are connected to the PSTN network and get the clock source from the ISDN over the ISDN port 2/3.
- We need four BRI ports, two for the PSTN and another two for the PBX. (Refer to section “Configuring ISDN Ports” on page 507).
- Furthermore we need four ISDN interfaces. Then we have to bind each BRI port to one of the ISDN interfaces. A hunt group that summarizes two ISDN interfaces is configured later during call router configuration.
- We need a call router routing table to route the calls depending on the called party number. (Refer to section “Configuring Call Routing” on page 505).
- We further need two hunt groups, one that hunts calls to the two BRI interfaces to the PSTN and one for the two BRI interfaces to the PBX.
- Then we need two other hunt group that tries to make a call over a VoIP and if this fails, falls back to the PSTN.
- We enable DTMF relay and specify codec G.723. (Refer to section “Configuring Voice Over IP Parameters” on page 506).

Configuring General CS Settings

First we set clock-source to ISDN port 2/3.

```
device>enable
device#configure
device(cfg)#system
device(sys)#clock-source 2 3
device(sys)#exit
device(cfg)#
```

Configuring Call Routing

Next we create the ISDN interfaces and configure call routing. Each interface is configured to route all incoming calls to the routing table TAB-CALLED-NUMBER. This table is part of the call router and configured below:

```
device(cfg)#context cs
device(ctx-cs)[switch]#interface isdn IF-PBX1
device(if-pstn)[IF-PBX1]#route call dest-table TAB-CALLED-NUMBER
device(if-pstn)[IF-PBX1]#exit

device(ctx-cs)[switch]#interface isdn IF-PBX2
device(if-pstn)[IF-PBX2]#route call dest-table TAB-CALLED-NUMBER
device(if-pstn)[IF-PBX2]#exit

device(ctx-cs)[switch]#interface isdn IF-PUBLIC-PSTN1
device(if-pstn)[IF-PUBL~]#route call dest-table TAB-CALLED-NUMBER
device(if-pstn)[IF-PUBL~]#exit

device(ctx-cs)[switch]#interface isdn IF-PUBLIC-PSTN2
device(if-pstn)[IF-PUBL~]#route call dest-table TAB-CALLED-NUMBER
device(if-pstn)[IF-PUBL~]#exit
device(ctx-cs)[switch]#
```

In addition, we create the two SIP interfaces and configure call routing, as well as the IP address of the remote SIP terminal, which is the IP address of the device in office A or office B, respectively.

```
device(ctx-cs)[switch]#interface sip IF-COMPOFF-A
device(if-sip)[IF-COMP~]#route call dest-table TAB-CALLED-NUMBER
device(if-sip)[IF-COMP~]#remoteip 146.86.130.11
device(if-sip)[IF-COMP~]#bind context sip-gateway SIP_GATEWAY
device(if-sip)[IF-COMP~]#exit

device(ctx-cs)[switch]#interface sip IF-COMPOFF-B
device(if-sip)[IF-COMP~]#route dest-table calledNumberRouting
device(if-sip)[IF-COMP~]#remoteip 146.86.130.24
device(if-sip)[IF-COMP~]#bind context sip-gateway SIP_GATEWAY
device(if-sip)[IF-COMP~]#exit
device(ctx-cs)[switch]#
```

Finally, we configure the call router. Here we create a routing table that examines the called party number of a call and routes numbers starting with a 1 and containing at least 3 digits to the hunt group that tries to reach company office A over VoIP and falls back to the PSTN. We route numbers starting with 2 and containing at least 3 digits to the hunt group that tries to reach company office B over VoIP and falls back to the PSTN. Calls

with a prefix of 5 and at least 3 digits are routed to the hunt group that selects a free BRI to the PBX and all other calls are routed to the hunt group that selects a free BRI to the PSTN:

```
device(ctx-cs)[switch]#routing-table called-e164 TAB-CALLED-NUMBER
device(rt-tab)[TAB-CAL~]#route 1.. dest-service HUNT-COMPOFF-A
device(rt-tab)[TAB-CAL~]#route 2.. dest-service HUNT-COMPOFF-B
device(rt-tab)[TAB-CAL~]#route 5.. dest-service HUNT-PBX
device(rt-tab)[TAB-CAL~]#route default dest-service HUNT-PUBLIC-PSTN
device(rt-tab)[TAB-CAL~]#show call-router config
Table switch/TAB-CALLED-NUMBER:
Key          Value          Function      Dest-Type      Dest-Name
called-e164  -
-----
1..          -              -            dest-service  HUNT-COMPOFF-A
2..          -              -            dest-service  HUNT-COMPOFF-B
5..          -              -            dest-service  HUNT-PBX
default     -              -            dest-service  HUNT-PUBLIC-PSTN
device(rt-tab)[TAB-CAL~]#exit
device(ctx-cs)[switch]#
```

The hunt group HUNT-COMPOFF-A tries to reach the company office A routing the call directly to the SIP interface IF-COMPOFF-A. When this call fails (e.g. because the data network is broken), we route the call to the PSTN hunt group. Likewise, hunt group HUNT-COMPOFF-B works, but tries to route the call to the SIP interface IF-COMPOFF-B first.

```
device(ctx-cs)[switch]#service hunt-group HUNT-COMPOFF-A
device(rt-tab)[HUNT-CO~]#no cyclic
device(rt-tab)[HUNT-CO~]#timeout 5
device(rt-tab)[HUNT-CO~]#route call 1 dest-interface IF-COMPOFF-A
device(rt-tab)[HUNT-CO~]#route call 2 dest-service HUNT-PUBLIC-PSTN
device(rt-tab)[HUNT-CO~]#exit
device(ctx-cs)[switch]#service hunt-group HUNT-COMPOFF-B
device(rt-tab)[HUNT-CO~]#no cyclic
device(rt-tab)[HUNT-CO~]#timeout 5
device(rt-tab)[HUNT-CO~]#route call 1 dest-interface IF-COMPOFF-B
device(rt-tab)[HUNT-CO~]#route call 2 dest-service HUNT-PUBLIC-PSTN
device(rt-tab)[HUNT-CO~]#exit
device(ctx-cs)[switch]#
```

The hunt group HUNT-PBX routes the call either to the interface IF-PBX1 or IF-PBX2, depending on which interface there is a free B channel. Likewise the hunt group HUNT-PUBLIC-PSTN works on the PSTN interfaces.

```
device(ctx-cs)[switch]#service hunt-group HUNT-PBX
device(rt-tab)[HUNT-PB~]#cyclic
device(rt-tab)[HUNT-PB~]#route call 1 dest-interface IF-PBX1
device(rt-tab)[HUNT-PB~]#route call 2 dest-interface IF-PBX2
device(rt-tab)[HUNT-PB~]#exit
device(ctx-cs)[switch]#service hunt-group HUNT-PUBLIC-PSTN
device(rt-tab)[HUNT-PU~]#cyclic
device(rt-tab)[HUNT-PU~]#route call 1 dest-interface IF-PUBLIC-PSTN1
device(rt-tab)[HUNT-PU~]#route call 2 dest-interface IF-PUBLIC-PSTN2
device(rt-tab)[HUNT-PU~]#exit
device(ctx-cs)[switch]#exit
device(cfg)#
```

Configuring VoIP Settings

Because we need G.723 as codec we enable DTMF relay:

```
device(cfg)#profile voip SIP-VOIP-PROFILE
device(pf-voip)[sip-VO~]#codec 1 g723-6k3
device(pf-voip)[sip-VO~]#dtmf-relay
device(pf-voip)[sip-VO~]#exit
device(cfg)#
```

We want to use this profile on our SIP interfaces:

```
device(cfg)#context cs
device(ctx-cs)[switch]#interface sip IF-COMPOFF-A
device(if-sip)[IF-COMP~]#use profile voip SIP-VOIP-PROFILE
device(if-sip)[IF-COMP~]#exit
device(ctx-cs)[switch]#interface sip IF-COMPOFF-B
device(if-sip)[IF-COMP~]#use profile voip SIP-VOIP-PROFILE
device(if-sip)[IF-COMP~]#exit
device(cfg)#
```

Configuring BRI Ports

Next step is to configure the BRI ports and to bind the ports to the ISDN interfaces. We configure the layer 2 (Q.921) to use point-to-point mode and layer 3 (Q.931) for user or net operation mode:

```
device(cfg)#port bri 2 0
device(prt-bri)[2/0]#q921
device(q921)[2/0]#protocol pp
device(q921)[2/0]#q931
device(q931)[2/0]#uni-side net
device(q931)[2/0]#bind interface IF-PBX1
device(q931)[2/0]#exit
device(q921)[2/0]#exit
device(prt-bri)[2/0]#no shutdown
device(cfg)#port bri 2 1
device(prt-bri)[2/1]#q921
device(q921)[2/1]#protocol pp
device(q921)[2/1]#q931
device(q931)[2/1]#uni-side net
device(q931)[2/1]#bind interface IF-PBX1
device(q931)[2/1]#exit
device(q921)[2/1]#exit
device(prt-bri)[2/1]#no shutdown
device(cfg)#port bri 2 2
device(prt-bri)[2/2]#q921
device(q921)[2/2]#protocol pp
device(q921)[2/2]#q931
device(q931)[2/2]#uni-side user
device(q931)[2/2]#bind interface IF-PBX1
device(q931)[2/2]#exit
device(q921)[2/2]#exit
device(prt-bri)[2/2]#no shutdown
device(cfg)#port bri 2 3
device(prt-bri)[2/3]#q921
device(q921)[2/3]#q931
device(q921)[2/3]#protocol pp
```

```

device(q931)[2/3]#uni-side user
device(q931)[2/3]#bind interface IF-PBX1
device(q931)[2/3]#exit
device(q921)[2/3]#exit
device(prt-bri)[2/3]#no shutdown

```

Configuring an SIP VoIP Connection

Next we configure call signaling:

```

device(cfg)#gateway sip sip
device(gw-sip)[sip]#no ras
device(gw-sip)[sip]#faststart
device(gw-sip)[sip]#bind interface eth0
device(gw-sip)[sip]#exit
device(cfg)#

```

Activating the CS Context Configuration

Prior to activating our configuration we use two **show** commands to display part of our configuration:

```

device(cfg)#show call-router config detail 5
Table switch/IF-PBX1-precallservice:
Key          Value          Function          Dest-Type          Dest-Name
-----
-            -              -                dest-table         TAB-CALLED-NUMBER

Table switch/IF-PBX2-precallservice:
Key          Value          Function          Dest-Type          Dest-Name
-----
-            -              -                dest-table         TAB-CALLED-NUMBER

Table switch/IF-PUBLIC-PSTN1-precallservice:
Key          Value          Function          Dest-Type          Dest-Name
-----
-            -              -                dest-table         TAB-CALLED-NUMBER

Table switch/IF-PUBLIC-PSTN2-precallservice:
Key          Value          Function          Dest-Type          Dest-Name
-----
-            -              -                dest-table         TAB-CALLED-NUMBER

Table switch/IF-COMPOFF-A-precallservice:
Key          Value          Function          Dest-Type          Dest-Name
-----
-            -              -                dest-table         TAB-CALLED-NUMBER

Table switch/IF-COMPOFF-B-precallservice:
Key          Value          Function          Dest-Type          Dest-Name
-----
-            -              -                dest-table         TAB-CALLED-NUMBER

```

```
Table switch/TAB-CALLED-NUMBER:
```

Key	Value	Function	Dest-Type	Dest-Name
called-e164	-			

1..	-	-	dest-service	HUNT-COMPOFF-A
2..	-	-	dest-service	HUNT-COMPOFF-B
5..	-	-	dest-service	HUNT-PBX
default	-	-	dest-service	HUNT-PUBLIC-PSTN

```
device(cfg)#
```

```
device(gw-sip)[gw_name]#exit
device(cfg)#debug cr full-detail
device(cfg)#context cs
device(ctx-cs)[switch]#no shutdown
02:30:26 CR > Updating tables in 3 seconds...
02:30:28 CR > [switch] Reloading tables now
02:30:28 CR > [switch] Flushing all tables
02:30:28 CR > [switch] Loading table 'IF-PBX1-precallservice'
02:30:28 CR > [switch] Loading table 'IF-PBX2-precallservice'
02:30:28 CR > [switch] Loading table 'IF-PUBLIC-PSTN1-precallservice'
02:30:28 CR > [switch] Loading table 'IF-PUBLIC-PSTN2-precallservice'
02:30:28 CR > [switch] Loading table 'IF-COMPOFF-A-precallservice'
02:30:28 CR > [switch] Loading table 'IF-COMPOFF-B-precallservice'
02:30:28 CR > [switch] Loading table 'TAB-CALLED-NUMBER'
device(ctx-cs)[switch]#
```

Showing the Running Configuration

The configuration script for our application looks as follows:

```
cli version 3.00

system
  clock-source 2 3

profile voip SIP-VOIP-PROFILE
  codec 1 g723-6k3 rx-length 30 tx-length 30
  codec 2 g711alaw64k rx-length 20 tx-length 20
  codec 3 g711ulaw64k rx-length 20 tx-length 20

context ip router

interface eth0
  ipaddress 147.86.130.1 255.255.225.0
  mtu 1500

interface eth1
  ipaddress 10.0.0.1 255.255.225.0
  mtu 1500

context cs switch

routing-table called-e164 TAB-CALLED-NUMBER
  route 1.. dest-service HUNT-COMPOFF-A
```



```
route 2.. dest-service HUNT-COMPOFF-B
route 5.. dest-service HUNT-PBX
route default dest-service HUNT-PUBLIC-PSTN

interface sip IF-COMPOFF-A
  bind context sip-gateway SIP_GATEWAY
  route call dest-table TAB-CALLED-NUMBER
  remote 146.86.130.11 5060
  privacy
  use profile voip SIP-VOIP-PROFILE

interface sip IF-COMPOFF-A
  bind context sip-gateway SIP_GATEWAY
  route call dest-table TAB-CALLED-NUMBER
  remote 146.86.130.24 5060
  privacy
  use profile voip SIP-VOIP-PROFILE

interface isdn IF-PBX1
  route call dest-table TAB-CALLED-NUMBER

interface isdn IF-PBX2
  route call dest-table TAB-CALLED-NUMBER

interface isdn IF-PUBLIC-PSTN1
  route call dest-table TAB-CALLED-NUMBER

interface isdn IF-PUBLIC-PSTN2
  route call dest-table TAB-CALLED-NUMBER

service hunt-group HUNT-COMPOFF-A
  timeout 5
  drop-cause normal-unspecified
  drop-cause no-circuit-channel-available
  drop-cause network-out-of-order
  drop-cause temporary-failure
  drop-cause switching-equipment-congestion
  drop-cause access-info-discarded
  drop-cause circuit-channel-not-available
  drop-cause resources-unavailable
  route call 1 dest-interface IF-COMPOFF-A
  route call 2 dest-service HUNT-PUBLIC-PSTN

service hunt-group HUNT-COMPOFF-B
  timeout 5
  drop-cause normal-unspecified
  drop-cause no-circuit-channel-available
  drop-cause network-out-of-order
  drop-cause temporary-failure
  drop-cause switching-equipment-congestion
  drop-cause access-info-discarded
  drop-cause circuit-channel-not-available
  drop-cause resources-unavailable
  route call 1 dest-interface IF-COMPOFF-B
  route call 2 dest-service HUNT-PUBLIC-PSTN
```

```
service hunt-group HUNT-PBX
  cyclic
  drop-cause normal-unspecified
  drop-cause no-circuit-channel-available
  drop-cause network-out-of-order
  drop-cause temporary-failure
  drop-cause switching-equipment-congestion
  drop-cause access-info-discarded
  drop-cause circuit-channel-not-available
  drop-cause resources-unavailable
  route call 1 dest-interface IF-PBX1
  route call 2 dest-interface IF-PBX2

service hunt-group HUNT-PUBLIC-PSTN
  cyclic
  drop-cause normal-unspecified
  drop-cause no-circuit-channel-available
  drop-cause network-out-of-order
  drop-cause temporary-failure
  drop-cause switching-equipment-congestion
  drop-cause access-info-discarded
  drop-cause circuit-channel-not-available
  drop-cause resources-unavailable
  route call 1 dest-interface IF-PUBLIC-PSTN1
  route call 2 dest-interface IF-PUBLIC-PSTN2

context cs switch
  no shutdown

context sip-gateway SIP_GATEWAY
  interface WAN_SIP
  bind interface eth0 context router port 5060
  context sip-gateway SIP_GATEWAY
  no shutdown

port ethernet 0 0
  medium 10 half
  encapsulation ip
  bind interface eth0 router
  no shutdown

port ethernet 0 1
  medium 10 half
  encapsulation ip
  bind interface eth1 router
  shutdown

port bri 2 0
  clock auto
  encapsulation q921

q921
  protocol pp
  uni-side auto
```

```
q931
  protocol dss1
  uni-side net
  bind interface IF-PBX1

port bri 2 0
  no shutdown

port bri 2 1
  clock auto
  encapsulation q921

q921
  protocol pp
  uni-side auto

q931
  protocol dss1
  uni-side net
  bind interface IF-PBX2

port bri 2 1
  no shutdown

port bri 2 2
  clock auto
  encapsulation q921

q921
  protocol pp
  uni-side auto

q931
  protocol dss1
  uni-side user
  bind interface IF-PUBLIC-PSTN1

port bri 2 2
  no shutdown

port bri 2 3
  clock auto
  encapsulation q921

q921
  protocol pp
  uni-side auto

q931
  protocol dss1
  uni-side user
  bind interface IF-PUBLIC-PSTN2
```

```
port bri 2 3  
no shutdown
```

Chapter 45 **CS Interface Configuration**

Chapter contents

Introduction.....	522
CS Interface Configuration Task List	522
Creating and Configuring CS Interfaces	523
Configuring Call Routing.....	524
Configuring the Interface Mapping Tables	525
Configuring the Precall Service Tables.....	528

Introduction

This chapter provides an overview of interfaces in the CS context and describes the tasks involved in their configuration. Within the CS context, an interface is a logical entity providing call signaling for incoming and outgoing calls to and from telephony ports and voice over IP gateways. It represents logical connections to other equipment or networks. CS interfaces are used as source and destination in the call router and are bound to physical ports or logical gateways.

Interface names can be any arbitrary string with a maximum of 25 characters. For ease of identification, the interface type can be a part of the name. [Figure 85](#) illustrates the function of the CS interfaces. The types of CS interfaces:

- PSTN interfaces telephony. Binding is done from a port to an interface.
- VoIP interface provide voice over IP settings in addition to the general CS interface parameters. These interfaces must be explicitly bound to an existing VoIP gateway.

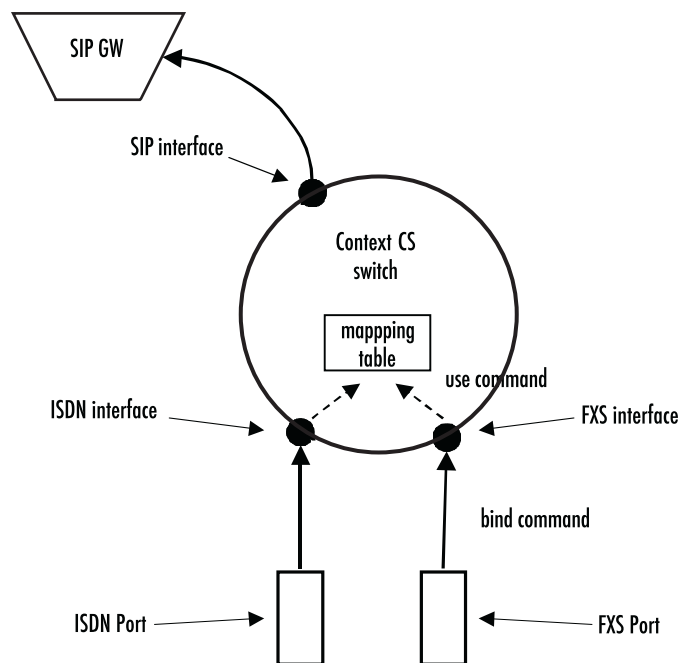


Figure 85. CS interfaces on the CS context

Interfaces can use mapping tables and precall service tables to manipulate call properties before the call is being offered to the call router.

CS Interface Configuration Task List

Several parameters depend upon the interface type. If it is not specifically stated otherwise, the configuration task is valid for all interfaces. This is not described in this chapter, but in Chapter 46, “[Tone Configuration](#)” on page 530 and chapter 42, “[VoIP Profile Configuration](#)” on page 470. To create and configure CS interfaces you have to perform the configuration tasks listed below.

- Creating and configuring CS interfaces

- Configuring call routing
- Configuring the interface mapping tables (optional)
- Configuring the precall service tables (optional)
- Configuring interface type specific parameters

Creating and Configuring CS Interfaces

To configure CS interfaces, you must first enter the CS context mode where you can create and configure your required interface through the CS interface configuration mode. Each interface has a name that can be any arbitrary string of not more than 25 characters. Use a name describing the purpose of the interface, as shown in the examples or—for ease of identification—the interface type can be used as part of the name. Already-defined CS interfaces can be displayed or deleted as described in the following table.

Procedure: Create and configure CS interfaces.

Mode: Configure

Step	Command	Purpose
1	node(cfg)#context cs	Enter the CS Context Configuration Mode.
2	node(ctx-cs)[switch]#interface if-type if-name	Enter the CS Interface Configuration Mode & select the CS interface with type <i>if-type</i> and name <i>if-name</i> for configuration. Valid interface types are sip, isdn, fxs and fxo.
3	node(if-type)[if-name]#...	Perform the configuration tasks to configure the CS interface.
4	node(ctx-cs)[switch]#show call-control provider	Display the configuration of the current CS interface.
5	node(if-type)[if-name]#exit	Go back to the CS Context Configuration Mode
6	node(ctx-cs)[switch]#show call-control provider OR node(ctx-cs)[switch]#show call-control status	Display already defined CS interfaces. Note: The show call-control provider command can also be used to display the configuration details of a provider either by specifying its name as a parameter or by being inside its configuration mode.
7	node(ctx-cs)[switch]#no interface if-type if-name	Delete an existing interface.

Examples: Create CS interfaces and delete another

The following example shows how to create and configure an interface, how to display it, and how to delete another.

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface isdn IF-PBX1
node(if-pstn)[IF-PBX1]#route call dest-interface TAB-CALLED-NUMBER
node(if-pstn)[IF-PBX1]#show call-control provider
Provider: IF-PBX1
=====
```

```

Binding: (none)
Protocol: (unknown)
DTMF Dialing: disabled
Tone-Set Profile: (none)
PSTN Profile: default
Routing Destination: router (IF-PBX1-precallservice)
Active Endpoints: 0
Suspended endpoints: 0
node(if-pstn)[IF-PBX1]#exit
node(ctx-cs)[switch]#show call-control provider
Call Control: switch
=====

Providers
-----

local
router
sn43
IF-PBX1
IF-PBX2
IF-PUBLIC-PSTN1
IF-PUBLIC-PSTN2
IF-COMPOFF-A
HUNT-COMPOFF-A
HUNT-PBX
HUNT-PUBLIC-PSTN
node(ctx-cs)[switch]#no interface isdn IF-PBX1
node(ctx-cs)[switch]#

```

Configuring Call Routing

Trinity offers two levels of call routing: basic interface routing and advanced call routing. Basic interface routing allows you to forward all incoming calls on a CS interface to a destination CS interface.

Advanced call routing allows you to route calls to all available CS interfaces, based on a criteria such as calling number, destination number, ISDN bearer capability, or other call properties. Using mapping tables, you can modify call properties like the calling or called party number, URI, etc. Furthermore, you can collect numbers using the digit-collection feature of called party number routing tables. Call services like hunt or distribution groups can be used to distribute calls to multiple destination interfaces.

In the environment of the CS interfaces, it is necessary to specify whether the call will be routed directly to another CS interface (basic interface routing) or to a first lookup table from the call router (advanced call routing).

In this chapter, only the configuration task on a CS interface is described. For configuration of the call routing tables, mapping tables and call services refer to Chapter 49, “[Call Router Configuration](#)” on page 561, which also describes the difference between the two levels of call routing in more detail.

Procedure: To configure basic interface routing

Mode: Context CS

Step	Command	Purpose
1	node(ctx-cs)[switch]#interface <i>if-type if-name</i>	Enters CS Interface Configuration Mode and configure interface <i>if-type</i> with name <i>if-name</i>
2	node(if-type)[if-name]#route call dest-interface <i>if-name</i> OR node(if-type)[if-name]#route call dest-table <i>table-name</i> OR node(if-type)[if-name]#route call dest-service <i>service-name</i>	Specifies a destination interface for incoming calls (basic interface routing) or a destination table or call service (advanced call routing)
3	node(if-type)[if-name]#exit	Returns to CS context configuration mode

Example: Configure call routing

The following example shows how to configure basic interface routing.

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface isdn IF-PBX1
node(if-pstn)[IF-PBX1]#route call dest-interface IF-SIP-0
node(if-pstn)[IF-PBX1]#exit
node(ctx-cs)[switch]#
```

Configuring the Interface Mapping Tables

Call router mapping tables are normally used by the call router to manipulate call properties during the call setup phase, i.e. when a call arrives on a CS interface and is routed to another interface through routing and mapping tables. This imposes a limitation to call property manipulation: When a call property like a party's number is changed during a call, the call is not routed through the call router again and thus, the mapping tables are not processed for the new number. Call property manipulation, e.g. removing a prefix from a number, cannot be done for the new number.

Consider, for example, an ISDN call, which may send a connected party number in the Connect message. This connected party number has the same meaning as the original called party number, but may differ from it. Another example of a call property that changes during a call is a SIP call transfer. A SIP call may be transferred to another user agent having a different URI than the called one. This new URI as well as the derived E.164 number cannot be manipulated using the call router before presenting it to the other party.

To circumvent this limitation, you can use mapping tables directly on an interface. In that case the mapping tables can be thought as input or output filters, which manipulate call properties at any stage of a call.

As with the SIP transfer example, differentiating *called* from *calling* party properties does not make sense for these manipulations, because the calling as well as the called party can be transferred in a SIP call. Therefore, mapping tables that are used on an interface manipulate both *called* and *calling* party properties at the same time! Although, using a calling-e164 mapping table at the interface will change the *calling* and *called* properties of the call.

You can chose different mapping tables for filtering parameters in each direction, input and output. While an input mapping table is applied to all properties that are received by the port or gateway that is bound to the interface before sending them to the peer interface in the CS context, an output mapping table is applied to all properties before sending them to the bound port or gateway.

Refer to the Chapter 49, “[Call Router Configuration](#)” on page 561 for more information about how to create and configure mapping tables.

Procedure: To use mapping tables to filter properties on an CS interface

Mode: Context CS

Step	Command	Purpose
1	node(ctx-cs)[switch]#interface <i>if-type if-name</i>	Enters CS Interface Configuration Mode and configure interface <i>if-type</i> with name <i>if-name</i>
2	node(if-type)[if-name]#use mapping-table in <i>table-name</i> AND/OR node(if-type)[if-name]#use mapping-table out <i>table-name</i>	Specifies an input and/or an output mapping table that shall be applied to all call properties in the specified direction.

Example: Use interface mapping tables for dialing plan conversion

The following example shows how to configure a dialing plan conversion on an interface. In this case, you can plan your call-routing tables to deal only with international numbers while converting private numbers on the CS interface that interfaces the private network.

```
node(ctx-cs)[switch]#mapping-table e164 to e164 PRIV-TO-GLOB
node(map-tab)[PRIV-TO~]#map (..) to 00419988825\1
node(map-tab)[PRIV-TO~]#exit
node(ctx-cs)[switch]#mapping-table e164 to e164 GLOB-TO-PRIV
node(map-tab)[GLOB-TO~]#map 00419988825(..) to \1
node(map-tab)[GLOB-TO~]#exit
node(ctx-cs)[switch]#interface isdn IF-PHONES
node(if-isdn)[IF-PHON~]#route call dest-table TAB-CALLED-NUMBER
node(if-isdn)[IF-PHON~]#use mapping-table in PRIV-TO-GLOB
node(if-isdn)[IF-PHON~]#use mapping-table out GLOB-TO-PRIV
node(if-isdn)[IF-PHON~]#exit
node(ctx-cs)[switch]#
```

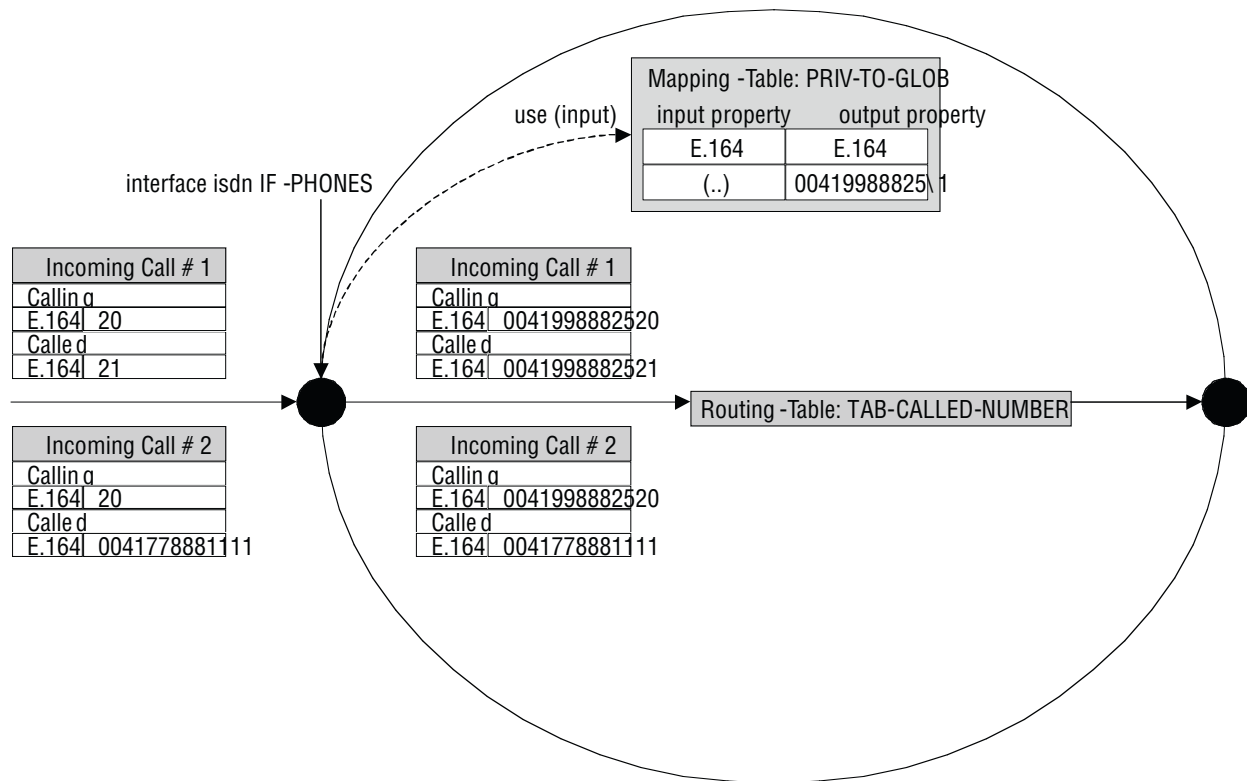


Figure 86. Incoming call passing an interface mapping table

Figure 86 shows two incoming calls arriving to the ISDN interface IF-PHONES. The calling and called party numbers are private numbers containing only two digits. Before accessing the call router, those numbers can be transformed into the global numbering plan. This is why the interface was configured to use mapping table PRIV-TO-GLOB on all incoming call properties.

Incoming call #1 originally has a calling party number of 20 and a called party number of 21. Before offering this call to the call router, mapping table PRIV-TO-GLOB is applied to the called party number and the calling party number. The mapping table adds a prefix of 00419988825 to the called and calling party number.

Incoming call #2 originally has a calling party number of 20 but already a called party number of the global numbering plan. Again, the mapping table is applied to both number, but only the calling party number of 20 is translated into 0041998882520. The called party number does not match an entry in the mapping table, so it is not changed.

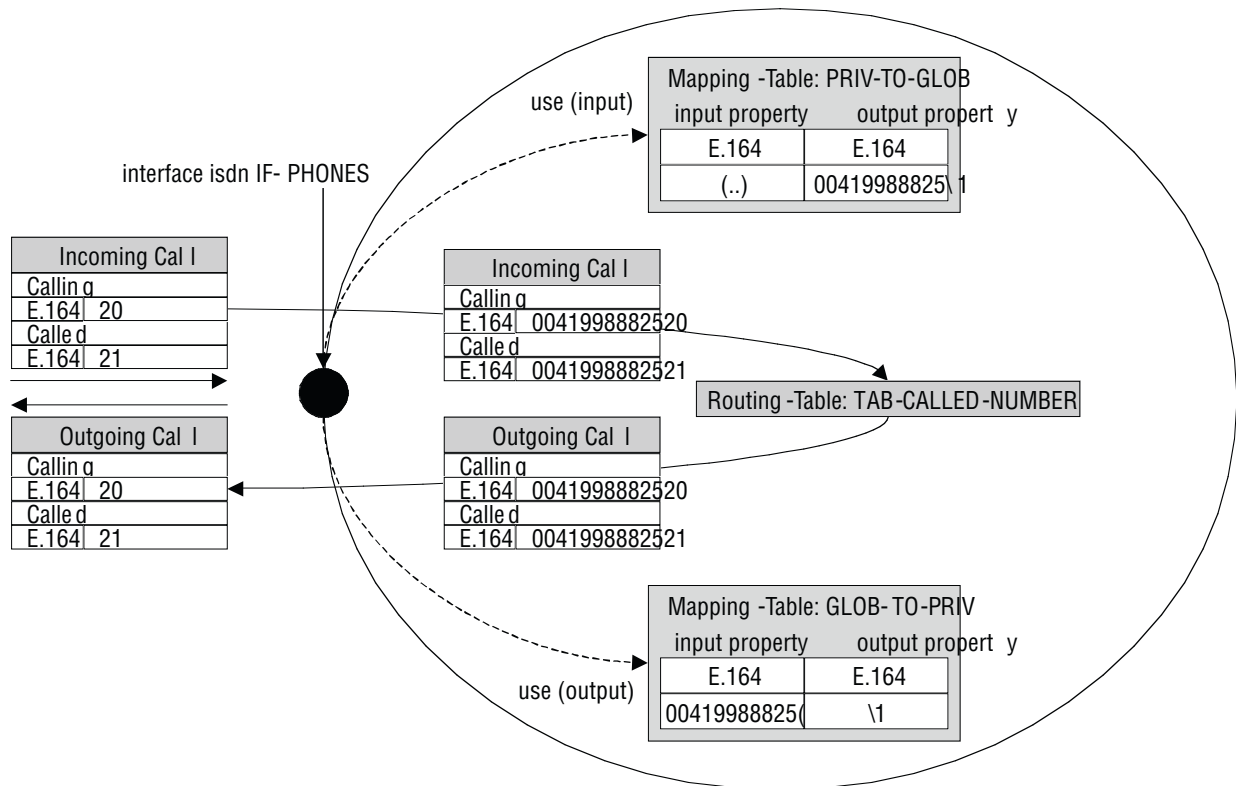


Figure 87. Call passing an input and an output mapping table

Let's assume we manipulate an incoming ISDN call using the PRIV-TO-GLOB mapping table as in the previous example. Figure 87 shows this situation again. Let's further assume the call router routes back the call to the interface IF-PHONES. In that case, the output mapping table used on this interface is applied to all call parameters. The calling and called party number is transformed from the global to the private numbering plan before the call is offered to the remote ISDN terminal.

Note For interface mapping you can use only mapping tables that examine general call parameters. For example, you cannot use a called-e164 to called-e164 mapping table, use a e164 to e164 mapping table instead.

Configuring the Precall Service Tables

Precall service tables are used to convert dialed special numbers such as *61 to activation commands for supplementary services such as call-waiting, etc. They are configured as part of the call router in the context CS mode and used on an FXS interface where the attached phone should be able to activate or deactivate services by dialing a special number. Trinity currently supports the following commands:

- **active-cw:** Activates call-waiting on the interface that uses the precall service table. Once activated a second call can be received on the same interface. The second call is announced to the subscriber who is engaged in the first call. The subscriber can decide whether to accept or reject the new call.
- **deactivate-cw:** Deactivates call-waiting on the interface that uses the precall service table.

- **interrogate-cw:** Detects whether or not the call-waiting supplementary service is active on the interface that uses the precall service table.

Procedure: To create a precall service table

Mode: Context CS

Step	Command	Purpose
1	<code>node(ctx-cs)[ctx-name]# [no] precall-service-table table-name</code>	Creates a new table that maps special numbers into supplementary service activation commands.
2	<code>node(ctx-cs)[ctx-name]# [no] map special-number to command</code>	Adds a new entry to map a special-number into a supplementary service activation, deactivation, or interrogation command.

Procedure: To use a precall service table on an FXS interface

Mode: Interface FXS

Command	Purpose
<code>node(if-fxs)[ctx-name.if-name]# [no] use mapping-table precall-service table-name</code>	Uses the precall service table <i>table-name</i> on the FXS interface <i>if-name</i> . That is, if the phone connected to the bound port dials a special number as configured in the used precall service table, the corresponding service is activated, deactivated, or interrogated.

Note Currently you can only use precall service tables on FXS interfaces

Example: Create and use a precall service table

The following example shows an example configuration snippet how to create a precall service table that treats *43# as activation command for the call-waiting supplementary service. The number #43# is used to deactivate call-waiting whereas *#43# is used to query the activation state of the service:

```
context cs SWITCH

precall-service-table PST-CALL-WAITING
  map *43# to activate-cw
  map #43# to deactivate-cs
  map *#43# to interrogate-cw

interface fxs IF-PHONE
  use mapping-table precall-service PST-CALL-WAITING
```

Chapter 46 **Tone Configuration**

Chapter contents

Introduction.....	531
Tone-set Profiles.....	531
Tone Configuration Task List	532
Configuring Call-Progress-Tone Profiles	532
Configure Tone-Set Profiles	533
Enable Tone-Set Profile	534
Show Call-Progress-Tone and Tone-Set Profiles	535

Introduction

This chapter gives an overview of call-progress-tone profiles and tone-set profiles, and describes the tasks involved in their configuration.

In-band tones keep the user informed about the state of his call or additional services such as call-waiting, hold etc. Other tones can be assigned to any event that occurs during a call, a call waiting tone, for example. The in-band tones are referred to as *call-progress-tones*.

Tone-set Profiles

In traditional PSTN networks the in-band tones (dial tone, alerting tone, busy tone etc.) are generated by the network, i.e. the Central Office switch or a similar device, and are relayed transparently by the Patton device. In voice over IP networks however this model of a network side providing services including in-band tones is not given in all situations. For example, two Patton devices may be connected directly to each other over the access network without the intervention of a traditional Central Office switch. This imposes the need to generate the local in-band tones directly on the gateways since none of the attached ISDN devices (PBXs, phones) will do so itself (ISDN USR side). The in-band tones that can be generated by the Patton device are the following:

- Busy tone—Tone you hear when you try to reach a remote extension but it is busy.
- Confirmation tone—Tone you hear when you enable a supplementary service and the system has accepted and activated it (for future use).
- Congestion tone—Tone you hear when you try to reach a remote extension but the network is busy or out of order (for future use).
- Dial tone—Tone you hear when you lift the handset and the network is ready to accept the dialed digits of the called party number.
- Hold tone—Tone you hear when you are in an active connection and the remote extension sets you ‘On Hold’ to reach a third party extension.
- Release tone—Tone you hear when you are in an active connection and the remote extension terminates the call.
- Ringback tone—Tone you hear when the called party number is complete and the remote extension is ringing.
- Special dial tone—Tone you hear when you lift the handset and the network is ready to accept the dialed digits of the called party number, but on your system is still an activated supplementary service (for future use).
- Special Information tone—Tone you hear when you try to reach a nonexistent remote extension (for future use).
- Waiting tone—Tone you hear when you already have an active connection and a second new extension tries to reach you.

All call-progress-tones are collected in a tone-set profile. A tone-set profile collects typically all required tones for one country. The tone-set profile is assigned to the PSTN interface (ISDN, FXS, FXO) or if it is required to have different tones for individual PSTN interfaces it's possible to assign for each PSTN interface its own tone-set profile. If no tone-set is assigned to a PSTN interface, the default tone-set is taken. [Figure 88](#) on page 532 illustrates the relationship between call-progress-tone profiles, tone-set profiles and PSTN interfaces.

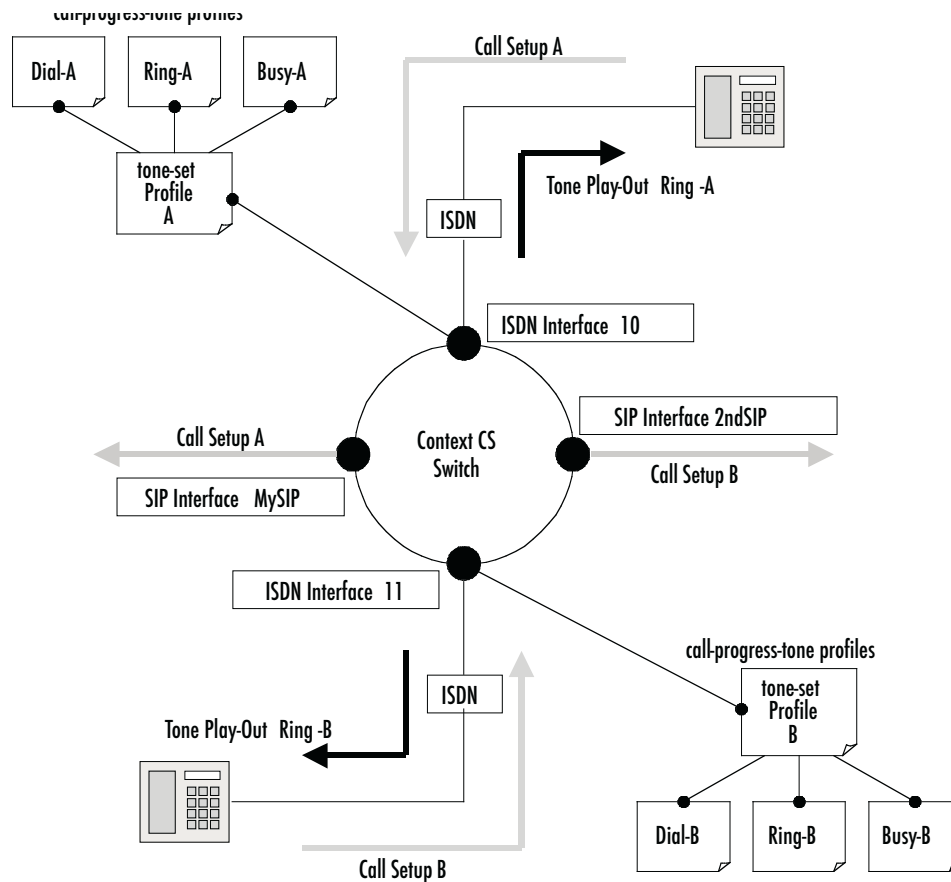


Figure 88. Assign tone-sets to a PSTN interfaces

Note There is a default tone-set named *default*, which maps the three Swiss standard in-band tones. Create a tone-set profile only if this default profile corresponds not with your country.

Tone Configuration Task List

To configure call progress tones, perform the tasks described in the following sections.

- Configuring call-progress-tone profiles
- Configuring tone-set profiles
- Enabling the generation of local in-band tones
- Showing call-progress-tone and tone-set profiles

Configuring Call-Progress-Tone Profiles

Each call-progress-tone consists of a sequence of different tones and pauses. Arbitrary tone cadences can be configured. All country specific tones can be defined with these parameters. Tone configuration knows only one command that has to be used repeatedly. The sequence in which the commands are entered (or appear in the config file) defines the sequence in which the corresponding elements are played.

Procedure: To configure a tone-set profile

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg)#profile call-progress-tone name</code>	Creates a call-progress-tone profile with name <i>name</i> and enters call-progress-tone configuration mode.
2	<code>node(pf-callp)[name]#play duration frequency1 level1 [frequency2 level2]</code>	Defines a tone with duration, frequency <i>frequency1</i> and volume <i>level1</i> . If a second frequency is defined both frequencies are played in parallel and for the same duration
3	<code>node(pf-callp)[name]#pause duration</code>	Defines a pause of <i>duration</i> milliseconds
4	<code>node(pf-callp)[name]#...</code>	Repeat step 2 and/or step 3 to define a tone sequence. Always when you enter a play or pause command, it is appended to the already existing tone.
5	<code>node(pf-callp)[name]#flush-play-list</code>	Resets the tone cadence. Same as deleting and re-creating the tone.

Example: Define the Belgian special information tone

The first line defines the first element of the tone: 330ms of 950Hz at -4dB. The second line the element that is played when the first element has finished: 330ms of 144Hz at -4dB, and so on. The last line defines a pause of 1 second after the three tones. The cadence is repeated infinitely.

```
node(cfg)#profile call-progress-tone belgianSpec
node(pf-callp)[belgian~]#play 330 950 -4
node(pf-callp)[belgian~]#play 330 1400 -4
node(pf-callp)[belgian~]#play 330 1800 -4
node(pf-callp)[belgian~]#pause 1000
```

Tones and pauses can be arbitrarily sequenced up to a number of 10 elements per call-progress-tone. The default call-progress-tone is an empty tone. The total number of different *play* elements across all configured call-progress-tones must not exceed 15 (an error is thrown if it does). If the call-progress-tone consists of only one element, this element has infinite duration. The *duration* parameter is ignored in this case.

Configure Tone-Set Profiles

A tone-set profile maps one call-progress-tone profile to each internal call-progress-tone. A tone-set profile typically includes all the call-progress-tones for one country.

Procedure: To configure a tone-set profile

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg)#profile tone-set name</code>	Creates tone-set <i>name</i> and enters tone-set profile configuration mode.

Step	Command	Purpose
2	node(pf-tones)[name]#map call_progress_tone { busy-tone confirmation-tone congestion-tone dial-tone hold-tone release-tone ringback-tone special-dial-tone special-information-tone waiting-tone } <i>call-progress-tone</i>	Map a call-progress-tone profile to an internal tone. An internal tone represents the call event for which a tone indication can be provided. Use the CLI help to get a list of all available events.
3		Repeat step 2 for all internal tone events.
4	[name](pf-tones)[name]#[dtmf-signal-level	Defines the output level of DTMF signals generated locally. This applies also to relayed DTMF signals.

Example: Configuring a tone-set

The following example shows how to configure a tone-set profile for UK.

```
node(cfg)#profile tone-set UK
node(pf-tones)[UK]#map call_progress_tone dialtone dialUK
node(pf-tones)[UK]#map call_progress_tone alertingtone ringUK
node(pf-tones)[UK]#map call_progress_tone busytone busyUK
```

Enable Tone-Set Profile

A call on the Patton device always has two signaling protocol endpoints. At the moment it is only possible to play locally generated tones on PSTN endpoints (ISDN, FXS) and not on IP based signaling endpoints (SIP). Dependent on the configuration several combinations of signaling protocol endpoints are possible (ISDN–ISDN, FXS-SIP etc.). The Patton device will always generate the tones locally and play it on the PSTN line as long as the other endpoint doesn't notify availability of in band information and the PSTN endpoint is NOT of type ISDN-USER or FXO. If availability of in band information will be notified by one endpoint, the bearer channel already contains the necessary tone information and must not be generated locally.

If the user has not specified a tone-set profile, the default tone-set will be taken to generate the local in band information. For enabling a user defined tone-set profiles on a specific interface proceed as follows.

Procedure: To assign a tone-set profile to a PSTN interface

Mode: Interface

Step	Command	Purpose
1	node(ctx-cs)[switch]#interface if-type if-name	Enter interface configuration mode.

Step	Command	Purpose
2	<code>node(if-type)[if-name]#use profile tone-set name</code>	Assign a user defined tone-set profile to an interface.

Example: Assign tone-set profiles to an ISDN interface

The example shows how to use the SWISS tone-set for the CS context, and use the USA tone-set for an individual interface.

```
node(cfg)#context cs
node(ctx-cs)[switch]#interface isdn bri0
node(if-isdn)[bri0]#use profile tone-set USA
```

Show Call-Progress-Tone and Tone-Set Profiles

Use the show commands to display the call-progress-tone profiles as well as the tone-set profiles.

Procedure: To show call-progress-tone profiles

Mode: Administrator execution

Step	Command	Purpose
1	<code>node#show profile call-progress-tone [name]</code>	Display all call-progress-tone profiles or a specific with a name

Example: Show call-progress-tone profile

The following example shows how to display the call-progress-tone profiles.

```
node#show profile call-progress-tone belgianSpec
Profiles:
-----

belgianSpec:
  Play 330ms (950Hz at -4dB)
  Play 330ms (1400Hz at -4dB)
  Play 330ms (1800Hz at -4dB)
  Pause 100ms
```

Procedure: To show tone-set profiles

Mode: Administrator execution

Step	Command	Purpose
1	<code>node#show profile tone-set [name]</code>	Display all tone-set profiles or a specific with name <i>name</i>

Example: Show tone-set profile

The following example shows how to display the tone-set profile.

```

node#show profile tone-set test
Tone Profile: test
=====

Used:                               by 0 module(s)
DTMF Duration:                       80ms
DTMF Interspace:                     80ms

Tones
-----

dial-tone:                           belgianSpec
ringback-tone:                       defaultAlertingtone
hold-tone:                           defaultHoldtone
waiting-tone:                         defaultWaitingtone
confirmation-tone:                   defaultConftone
busy-tone:                            defaultBusytone
congestion-tone:                     defaultCongestiontone
release-tone:                         defaultReleasetone
special-information-tone:             defaultSITone
special-dial-tone:                   defaultSDtone

```

Example: The following example shows how to configure a tone-set profile for UK and apply it to the isdn interface bri0.

Create the call-progress-tone profiles:

```

node(cfg)#profile call-progress-tone dial-UK
node(pf-callp)[dial-UK]#play 5000 350 0 440 0

node(pf-callp)[dial-UK]#profile call-progress-tone alerting-UK
node(pf-callp)[alertin~]#play 400 400 0 450 0
node(pf-callp)[alertin~]#pause 200
node(pf-callp)[alertin~]#play 400 400 0 450 0
node(pf-callp)[alertin~]#pause 2000

node(pf-callp)[alertin~]#profile call-progress-tone busy-UK
node(pf-callp)[busy-UK]#play 400 400 0
node(pf-callp)[busy-UK]#pause 400
node(pf-callp)[busy-UK]#exit

```

Create the tone-set profile:

```

node(cfg)#profile tone-set UK
node(pf-tones)[UK]#map call_progress_tone dialtone dial-UK
node(pf-tones)[UK]#map call_progress_tone alertingtone alerting-UK
node(pf-tones)[UK]#map call_progress_tone busytone busy-UK
node(pf-tones)[UK]#exit

```

Assign the tone-set to the isdn interface bri0

```

node(cfg)#context cs
node(ctx-cs)[switch]#interface isdn bri0
node(if-isdn)[bri0]#use profile tone-set UK

```

Chapter 47 **Authentication Service**

Chapter contents

Introduction	538
Authentication Service Configuration Task List	538
Creating an Authentication Service	538
Configuring a Realm	539
Configuring the Authentication Protocol	539
Creating Credentials	539
Configuration Examples	539

Introduction

This chapter describes how to configure authentication services in Trinity. The *Authentication Service* is a data base that manages *Authentication Credentials* of one or more *Realm*. A *Realm* is an *Authentication Zone* or *Authentication Domain* that defines the authentication responsibility in a network. Each *Authentication Credential* created in an Authentication Service belongs to the defined Realm and exists on a User Name and an optional Password. It is also possible to create an Authentication Service without specifying a Realm, which represents the default realm. Whenever authentication is required and the provided Realm doesn't exist in an Authentication Service, this default realm will be considered to find the right Authentication Credentials.

Authentication Service Configuration Task List

The following section describes how to create a new authentication service and how to enter the configuration mode of an existing service. Additionally, it describes all commands and sub commands of the authentication service configuration mode. All configuration tasks for Authentication Services are listed below.

- Create an Authentication Service (see page 538)
- Configure a Realm (see page 539)
- Configure the authentication protocol (see page 539)
- Create credentials (see page 539)

Creating an Authentication Service

The **authentication-service** command enters the configuration mode of an existing authentication or creates a new one if the requested service does not yet exist. The *no* form of the command destroys the authentication service.

Mode: Configure

Step	Command	Purpose
1	<code>[node](cfg)# [no] authentication-service <name></code>	Creates/Destroys an authentication service or enters existing authentication-service configuration mode.

Configuring a Realm

The following commands add a new Realm to the authentication service. If more than one Realm has to be entered, the order of the list can be modified by using the *index* and/or *before* and *after* keywords. The no form of the command removes an existing Realm from the list.

Mode: Authentication Service

Step	Command	Purpose
1	<code>[node](Is)[name]# [no] realm <name></code> OR <code>[node](Is)[name]#realm <index> <name></code> OR <code>[node](Is)[name]#realm before <index> <name></code> OR <code>[node](Is)[name]#realm after <index> <name></code>	Adds or removes a Realm to/from the authentication service.

Configuring the Authentication Protocol

The **protocol** command specifies the protocol.

Mode: Authentication Service

Step	Command	Purpose
1	<code>[node](Is)[name]#protocol {http}</code>	Specifies the authentication protocol to be used.

Creating Credentials

The following command creates Authentication Credentials identified by the entered username. The no form of the command removes an existing Credential. It is possible to enter this command without a password.

Mode: Authentication Service

Step	Command	Purpose
1	<code>[node](Is)[name]# [no] username <user></code> <code>password <password></code>	Creates or removes authentication credentials.

Configuration Examples

```
authentication-service AUTH_SRV
  realm 1 voip-public
  realm 2 voip-intranet
  realm 3 ms-exchange
  username 433 password fK+bfnzL45Goh/VdjrWxAA== encrypted
  username john.doe password D60t7CBZ58k7JK2jxdlw4w== encrypted
```

Chapter 48 Location Service

Chapter contents

Introduction	541
Location Service Configuration Task List	541
Creating a Location Service	541
Adding a Domain	541
Configuring Default Responsibility	542
Creating an Identity	542
Authentication outbound face	544
Authentication inbound face	545
Registration outbound face	546
Registration Priority	548
DNS Security Check	548
Support broken SIP proxy	549
NAT-traversal 'flows'	549
SIP B2BUA Dynamic Registration	550
Registration inbound face	550
Call outbound face	551
Configuring the Dynamic Registrar Use	552
Support broken SIP proxy	553
Call inbound face	554
Configuring SIP Transaction Timeout and Penalty Box	554
Creating an Identity Group	555
Inheriting from an Identity Group to an Identity	555
Configuring the Message Waiting Indication Feature for SIP	556
Subscription	556
Notification	557
Configuration	557
Message Waiting Indication through Call-Control	559
Show Location-Service	559
Configuration Examples	559

Introduction

This chapter describes how to configure location services in Trinity.

Location Service Configuration Task List

The following section describes how to create a new location service and how to enter the configuration mode of an existing service. Additionally, it describes all commands and sub commands of the location service configuration mode. All configuration tasks for Location Services are listed below:

- Create a Location Service (see page 541)
- Add a domain (see page 541)
- Create an identity (see page 542)
- Create an identity group (see page 555)
- Inherit from an identity group to an identity (see page 555)

Creating a Location Service

The **location-service** command enters the configuration mode of a location service. If the requested service does not yet exist, a new one will be created. The no form of the command removes an existing location service.

Mode: Configure

Step	Command	Purpose
1	<code>[device](cfg)# [no] location-service <name></code>	Creates/Destroys a location service or enters configuration mode.

Adding a Domain

The **domain** command specifies the domains that the location service is responsible for. If the application needs information from the location service, it performs a lookup with the Host Part of the Request-URI or the From-URI to find the right instance. The header selection from which the URI will be taken depends on the call direction (Outgoing/Incoming SIP-Call) and the requested information. The SIP environment determines which format the Domain has; it can either be a Domain-Name, a Host-Name or a Host-Address. If all components of the SIP environment are set up to operate in one specified domain, Domain is a Domain-Name. If point-to-point routing is used, Domain is either a Host-Name or a Host-Address. In this case, Host-Name is a FQDN (Full Qualified Domain Name).

Domain Examples:

Domain-Name: *biloxi.com*

Host-Name: *sip-ua.biloxi.com* or *sip-server.biloxi.com*

Host-Address: *192.168.10.1* or *10.10.10.1*

In case of point-to-point routing, local host-addresses may not be added to the domain list of a location service; these addresses are known by the application. But, if an identity exists in two different location services and the

Context SIP Gateway has more than one transport binding, it is recommended to add the local host addresses as *Domain* to the appropriated location services.

Mode: Location Service

Step	Command	Purpose
1	[device](svc-ls)[ls]# [no] domain <name> OR [device](svc-ls)[ls]#domain <index> <name> OR [device](svc-ls)[ls]#domain before <index> <name> OR [device](svc-ls)[ls]#domain after <index> <name>	Adds a new domain to the location service. If more than one domain has to be entered, the order of the list can be modified by using the index and/or the insert keywords <i>before</i> and <i>after</i> . The no form of the command removes an existing domain from the list.

Configuring Default Responsibility

A location service can be configured to accept responsibility for any domain. This eases the matching of dynamic ip addresses into the location service. When the Sip gateway does a lookup in the database it considers the location service where a configured domain matches the host part of the identity in the first priority. If no location services are found, the first location service which is responsible for any domain is taken.

Mode: Location Service

Step	Command	Purpose
1	[device](svc-ls)[ls]#[no] match-any-domain	Adds or removes the responsibility for any possible domain or ip address.

Creating an Identity

An identity represents one of multiple possible addresses over which a user is reachable (e.g. sip:john@patton.com). This leads to a huge range of configuration possibilities in the identity.

According to the relationship between an identity and user, there can be many different aspects configured. If you are the user agent for a certain identity, use the outbound faces to specify the behavior when sending requests. If you are not the user agent of an identity but know this identity, then use the inbound faces to configure the behavior when this identity sends requests.

When creating an identity, it is important to consider that the name of the identity is always used as user-part when building a sip-uri. The name of the identity is also used when comparing to or matching with a sip-uri.

Mode: Location Service

Step	Command	Purpose
1	[device](svc-ls)[ls]#[no] identity <name>	Adds a new identity to the location service. The no form of the command removes an existing identity.

Mode: Identity

Step	Command	Purpose
1	<code>[device](identity)[device]# [no] display-name <display-name></code>	Adds a display-name to the identity. The no form removes the display-name.

Mode: Identity

Step	Command	Purpose
1	<code>[device](identity)[device]# [no] phone-context <phone-context></code>	Adds a phone-context to the identity. The no form removes the phone-context.

An alias is an alternative way to express the user-part of an identity. An alias is never used to build a sip-uri and will never be used in communication with another device. The alias is used for comparing or matching the identity with a sip-uri received from an external device.

Mode: Identity

Step	Command	Purpose
1	<code>[device](identity)[device]# [no] alias name <alias></code> OR <code>[device](identity)[device]# [no] alias range <start><end></code> OR <code>[device](identity)[device]# [no] alias expression <expression></code>	Adds a new alias to the identity. The no form of the command removes an existing alias. [no] removes the alias again. Range values must be numerical and the start value must be smaller than the end value.
2	<code>[device](identity)[device]# alias none</code>	Clears the entire alias list for this identity.

RFC3261 defines a user-param for SIP URIs. The value of this parameter can be configured for identities in the location service. The configured value will then be applied to any SIP URI for which the user part matches the identity.

- **phone:** This identity represents a telephone number. This is typically used when a phone-context has been configured.
- **dialstring:** This identity represents a dialstring.
- **ip:** This identity represents an ordinary SIP user. This is the default value assumed by SIP also if no user-param is specified. Thus it usually does not make any difference whether “user ip” or “no user” is specified even though the generated SIP URIs will be different.

Mode: Identify

Step	Command	Purpose
1	<code>[device](identity)[device]# [no] user {phone dialstring ip}</code>	Defines the value of the user-param in SIP URIs where this identity is present.

The huge amount of possible configuration parameters has been separated into different configuration entities called the faces. The faces refer to authentication, registration and call. Each face works in two directions: inbound and outbound. Outbound faces refer to requests originating from your identity. Inbound faces refer to requests destined to your identity.

- Authentication outbound face (see page 544)
- Authentication inbound face (see page 545)
- Registration outbound face (see page 546)
- Registration inbound face (see page 550)
- Call outbound face (see page 551)
- Call inbound face (see page 553)

Authentication outbound face

The authentication outbound face is used to provide authentication credentials to challenges from other user agents or proxies.

Mode: Identity

Step	Command	Purpose
1	<code>[device](identity)[device]# [no] authentication outbound</code>	Adds a new face to the identity. The no form of the command removes an existing face with all content in it.

An authentication entry establishes a link between an identity and exactly one pair of credentials in an authentication-service. To link multiple credentials to an identity, there must be one authentication entry in the authentication outbound face for each pair of credentials to link. The parameter username selects the entry in the authentication-service to use. The parameter username can be omitted if and only if the name of the identity matches exactly the username in the authentication-service.

Mode: Authentication outbound

Step	Command	Purpose
1	<pre>[device](authout)#authenticate authentication-service <authentication-service> [username <username>] OR [device](authout)#authenticate <index> authentication-service <authentication-service> [username <username>] OR [device](authout)#authenticate before <index> authentication-service <authentication-service> [username <username>] OR [device](authout)#authenticate after <index> authentication-service <authentication-service> [username <username>]</pre>	Adds a new authentication entry to the authentication outbound face. If more than one authentication entry has to be entered, the order of the list can be modified by using the index and/or the insert keywords before and after.

Mode: Authentication outbound

Step	Command	Purpose
1	<pre>[device](authout)#no authenticate [<index>]</pre>	Removes the authentication entry at the index or removes all authentication entries if no index is given.

Mode: Authentication outbound

Step	Command	Purpose
1	<pre>[device](authout)#authenticate none</pre>	Removes all authentication entries and disables explicitly authentication outbound.

Authentication inbound face

The authentication inbound face is used when you want to challenge other user agents.

Mode: Identity

Step	Command	Purpose
1	<pre>[device](identity)[device]# [no] authentication inbound</pre>	Adds a new face to the identity. The no form of the command removes an existing face with all content in it.

Mode: Authentication inbound

Step	Command	Purpose
1	<pre>[device](authin)#authenticate authentication-service <authentication-service> [username <username>] OR [device](authin)#authenticate <index> authentication-service <authentication-service> [username <username>] OR [device](authin)#authenticate before <index> authentication-service <authentication-service> [username <username>] OR [device](authin)#authenticate after <index> authentication-service <authentication-service> [username <username>]</pre>	Adds a new authentication entry to the authentication inbound face. If more than one authentication entry has to be entered, the order of the list can be modified by using the index and/or the insert keywords before and after.

Mode: Authentication inbound

Step	Command	Purpose
1	<pre>[device](authin)#no authenticate [<index>]</pre>	Removes the authentication entry at the index or removes all authentication entries if no index is given.

Mode: Authentication inbound

Step	Command	Purpose
1	<pre>[device](authin)#authenticate none</pre>	Removes all authentication entries and disables explicitly authentication inbound.

Registration outbound face

The registration outbound face is used to register an identity on an external registrar. Then, the registrar forwards calls from the registered identity to your identity.

Mode: Identity

Step	Command	Purpose
1	<pre>[device](identity)[device]# [no] registration outbound</pre>	Adds a new face to the identity. The no form of the command removes an existing face with all content in it.

Mode: Registration outbound

Step	Command	Purpose
1	<code>device(regout)# [no] register [auto none back-to-back]</code>	Enables registration with auto or disables registration explicitly with none. For the back-to-back option, refer to section “SIP B2BUA Dynamic Registration” on page 550.

Mode: Registration outbound

Step	Command	Purpose
1	<code>[device](regout)# [no] registrar <host> [<port>]</code>	Configures the address of the registrar to send your register requests.

Mode: Registration outbound

Step	Command	Purpose
1	<code>[device](regout)# [no] lifetime <seconds></code>	Configures the desired lifetime of the registration. When no lifetime is configured the desired lifetime is set to 3600 seconds.

Mode: Registration outbound

Step	Command	Purpose
1	<code>[device](regout)# [no] retry-timeout (on-client-error on-system-error on-server-error) <seconds></code>	Configures the time to wait after a failed registration according to three different error categories. After this time we begin to register from new. If no retry-timeout is configured the retry-timeout is set to 10 seconds.

Mode: Registration outbound

Step	Command	Purpose
1	<code>[device](regout)#proxy <host> [<port>] [strict-route]</code> OR <code>[device](regout)#proxy <index> <host> [<port>] [strict-route]</code> OR <code>[device](regout)#proxy before <index> <host> [<port>] [strict-route]</code> OR <code>[device](regout)#proxy after <index> <host> [<port>] [strict-route]</code>	Adds a new proxy entry to the registration outbound face. For each proxy configured there is a route-header added. If more than one proxy entry has to be entered, the order of the list can be modified by using the index and/or the insert keywords before and after.

Mode: Registration outbound

Step	Command	Purpose
1	[<i>device</i>](regout)#proxy <index> down <positions> OR [<i>device</i>](regout)#proxy <index> up <positions>	If multiple proxies are configured the entry at the index can be moved in the proxy list up or down the number of positions given in the command.

Mode: Registration outbound

Step	Command	Purpose
1	[<i>device</i>](regout)#no proxy [<index>]	Removes the proxy entry at the index or remove all proxy entries if no index is given.

Mode: Registration outbound

Step	Command	Purpose
1	[<i>device</i>](regout)#proxy none	Removes all proxy entries and disables explicitly the use of a proxy.

Mode: Registration outbound

Step	Command	Purpose
1	[<i>device</i>](regout)#[no] preferred-transport-protocol (tcp udp)	Selects the preferred transport protocol for conveying the REGISTER request.

Registration Priority. The *q* parameter is a value between 0 and 1 and specifies the weight of a REGISTER message. It is mainly used to control the call handling priority and sequence in forking applications with support for multiple registrations of the same user.

The default value of 1 means that the *q* value will not be added to the contact header. By default, the *q* value is disabled.

Mode: Registration outbound

Step	Command	Purpose
1	[<i>device</i>](regout)#[no] priority <priority>	Specifies the weight of the registration in per mille (Example: 800 -> q='0.8'). Default = Disabled

DNS Security Check. Upon receipt of a redirection response (a 300, 301 or 302 SIP response), the SIP User Agent formulates a new REGISTER request based on the request specified in the Contact field. It is possible to configure to register using the contact only if it matches one of the addresses previously received from the DNS resolution.

Mode: Registration outbound

Step	Command	Purpose
1	<code>[device](regout)#[no] check-dns-addresses</code>	Use only registrar pertaining to addresses previously received from DNS resolution.

Example: Enable DNS Security Check

```
location-service LOC
domain 1 example.com

identity 100

registration outbound
check-dns-addresses
register auto
```

Support broken SIP proxy. To direct SIP requests to outbound proxies without having a proxy header in the message a new command is introduced. This command specifies a destination host (with optional port) to which SIP messages will be sent regardless of domain, proxies and registrar.

Note This feature breaks RFC 3261! Use this feature only if you are certain that the addressed behaves as expected and is able to process such messages.

Mode: Registration outbound

Step	Command	Purpose
1	<code>[device](regout)#[no] force-destination [host [port]]</code>	Specifies a host and optional port to send registration messages, overriding proxies, registrar and domain.

NAT-traversal 'flows'

In the SIP location service, the NAT-traversal 'flows' feature was not supported. Now, according to RFC5626, it is possible to configure NAT-traversal flows in the registration outbound mode within a location-service. This allows the device to receive incoming calls on the same flow (UDP, TCP, or TLS connection) used for outbound registration. This feature is helpful to overcome NAT gateways: If the Patton device is located in a private network and registers to a public SIP registrar/switch, the calls from the switch to the device will re-use the established registration connection that passed through the NAT gateway already. Optionally, outbound calls can also be sent through the registration flow, which has to be configured in the 'call outbound' mode.

In addition, a new command 'nat-traversal minimal' has been added and the existing command 'nat-traversal keep-alive' has been modified to have the possibility to choose between the keep alive methods 'options' or 'ping'.

Mode: location-service

Step	Command	Purpose
1	node(regout)# nat-traversal flows (crlf no-keep-alive options ping) [keepalive-interval <seconds>]	Enable NAT traversal with flows according to RFC5626 where inbound calls may reach the device over an established registration connection (flow); this works for all transport protocols: UDP, TCP, and TLS.
2	node(regout)# nat-traversal minimal	Enable NAT traversal without keep alive messages and adapt the contact header to public parameters.
3	node(regout)# nat-traversal keep-alive (options ping) [keepalive-interval <seconds>]	Enable NAT traversal by sending OPTIONS or PING requests as keep alive messages.

SIP B2BUA Dynamic Registration

This feature gives the possibility to trigger an outbound registration when a matching inbound registration has been detected. As prerequisite you will need to define an inbound registration for the desired identity in the location service. You will need to define a registrar in the outbound registration configuration as well.

Mode: location-service

Step	Command	Purpose
1	device(ls)[name]#[no] identity <name>	Adds a new identity to the location service.
2	device(identity)# [no] authentication inbound	Define if inbound authentication is used or not.
3	device(identity)# [no] authentication outbound	Define if outbound authentication is used or not.
4	device(identity)[name]# [no] registration inbound	Add inbound registration
5	device(identity)[name]# [no] registration outbound	Add outbound registration
6	device(regout)# [no] register [auto none back-to-back]	Set the back-to-back parameter in the outbound registration configuration

Registration inbound face

The registration inbound face is used when you want to allow external user agents to register, so that you can route calls to the registered contacts.

Mode: Identity

Step	Command	Purpose
1	[device](identity)[device]# [no] registration inbound	Adds a new face to the identity. The no form of the command removes an existing face with all content in it.

Mode: Registration inbound

Step	Command	Purpose
1	<code>[device](regin)# [no] lifetime [default <seconds>] [min <seconds>] [max <seconds>]</code>	Configures the range of the expiration time accepted for inbound registration. If there is a time requested which is out of the range the time is set to a value which fits the range. If there is no time requested it is set to the default lifetime.

Mode: Registration inbound

Step	Command	Purpose
1	<code>[device](regin)# [no] contact <host> [<port>] <context cs> <interface sip> [priority <priority>]</code>	Adds a contact to the registration inbound face. The interface and context parameter defines where the sip-location-service routes a call destined to that contact. The higher priority a contact has the sooner a contact is chosen. The default priority is 1000 which is also the maximum priority. The no form of the command removes a contact.

Call outbound face

The call outbound face is used to configure call properties for outgoing calls.

Mode: Identity

Step	Command	Purpose
1	<code>[device](identity)[device]# [no] call outbound</code>	Adds a new face to the identity. The no form of the command removes an existing face with all content in it.

Mode: Call outbound

Step	Command	Purpose
1	<code>[device](callout)device# [no] use profile (sip tone-set voip) <profile></code>	Adds or removes a profile to use if an outgoing call destines this identity

Mode: Call outbound

Step	Command	Purpose
1	<code>[device](callout)device# [no] preferred-transport-protocol (tcp udp)</code>	Selects which protocol to prefer if an outgoing call destines this identity.

Mode: Call outbound

Step	Command	Purpose
1	<code>[device](callout)device# [no] traffic-class <traffic-class></code>	Selects which traffic class to set if an outgoing call destines this identity.

Mode: Call outbound

Step	Command	Purpose
1	<code>[device](callout)#proxy <host> [<port>] [strict-route]</code> OR <code>[device](callout)#proxy <index> <host> [<port>] [strict-route]</code> OR <code>[device](callout)#proxy before <index> <host> [<port>] [strict-route]</code> OR <code>[device](callout)#proxy after <index> <host> [<port>] [strict-route]</code>	Adds a new proxy entry to the call outbound face. If the from-uri of a call originating from us matches the identity for each proxy configured there is a route-header added. If more than one proxy entry has to be entered, the order of the list can be modified by using the index and/or the insert keywords before and after.

Mode: Call outbound

Step	Command	Purpose
1	<code>[device](callout)#proxy <index> down <positions></code> OR <code>[device](callout)#proxy <index> up <positions></code>	If multiple proxies are configured the entry at the index can be moved in the proxy list up or down the number of positions given in the command.

Mode: Call outbound

Step	Command	Purpose
1	<code>[device](callout)#no proxy [<index>]</code>	Removes the proxy entry at the index or remove all proxy entries if no index is given.

Mode: Call outbound

Step	Command	Purpose
1	<code>[device](callout)#proxy none</code>	Removes all proxy entries and disables explicitly the use of a proxy.

Configuring the Dynamic Registrar Use. Upon receipt of a redirection response (a 300, 301 or 302 SIP response), the SIP User Agent formulates a new REGISTER request based on the request specified in the Contact field. It is possible to configure that all the subsequent SIP requests be directly sent to the redirected registrar if the REGISTER succeeded.

Mode: Call outbound

Step	Command	Purpose
1	<code>[device](callout)#[no] request-uri dynamic-registrar</code>	Enable the use of the dynamic registrar to direct the request. Default: Disabled.

Example: Enable the Use of the Dynamic Registrar

```
location-service LOC
domain 1 example.com

identity 100

call outbound
request-uri dynamic-registrar
```

Support broken SIP proxy. To direct SIP requests to outbound proxies without having a proxy header in the message a new command is introduced. This command specifies a destination host (with optional port) to which SIP messages will be sent regardless of domain, proxies and registrar.

Note This feature breaks RFC 3261! Use this feature only if you are certain that the addressed behaves as expected and is able to process such messages.

Mode: Call outbound

Step	Command	Purpose
1	<code>device(callout)#[no] force-destination manual [host [port]]</code>	Specifies a host and optional port to send INVITE messages, overriding proxies, registrar and domain.
2	<code>node(callout)#[no] force-destination registrar address</code>	Allows sending INVITE requests to the IP Address of the registrar where a given identity is currently registered to. For this purpose the destination IP address is forced to the detected registrar address independently of any URLs present in any SIP header of the INVITE packet.

Calls using flows. The concept of flows can be used for calls as well. When flows are opened by registrations, these are reused for outgoing calls, but only if the target resolution procedures for calls results in an IP address for which a flow already exists. Otherwise, a new flow is created for the duration of the call.

This does not affect incoming calls. Incoming calls are accepted through flows or outside of flows in the same way.

Mode: Call outbound

Step	Command	Purpose
1	<code>[device](callout)# [no] nat-traversal flows</code>	Configured if INVITE requests are sent out through flows.

Call inbound face

The call inbound face is used to configure call properties for incoming calls.

Mode: Identity

Step	Command	Purpose
1	<code>[device](identity)[device]# [no] call inbound</code>	Adds a new face to the identity. The no form of the command removes an existing face with all content in it.

Mode: Call inbound

Step	Command	Purpose
1	<code>[device](callin)device# [no] use profile (sip tone-set voip) <profile></code>	Adds or removes a profile to use if an incoming call destines this identity.

Configuring SIP Transaction Timeout and Penalty Box

The transaction timeout defines how long the SIP gateway tries to establish a transaction to a certain destination without success before the transaction is considered as failed and a possible alternative destination is tried. This is the case when a DNS-lookup returns multiple ip-addresses for a destination. Therefore, the invite transaction timeout is taken for INVITE transactions and the non-invite transaction timeout for any other transaction.

When the penalty box is used, such a failed destination is put into the penalty box for 5 minutes. Any destination which is not in the penalty box has a higher priority for use as destinations listed in the penalty box.

For outgoing calls, the invite-transaction-timeout, the non-invite-transaction-timeout and the use of the penalty box is taken from the Identity corresponding to the request-uri or the identity-group “default” in the location service corresponding to the host part of the request-uri.

For incoming calls, the invite-transaction-timeout, the non-invite-transaction-timeout and the use of the penalty box is taken from the Identity corresponding to the from-uri or the identity-group “default” in the location service corresponding to the host part of the from-uri.

Mode: Identity/Call outbound

Step	Command	Purpose
1	<code>[device](callout)# [no] invite-transaction-timeout <seconds></code>	Sets the transaction timeout for invite transactions directed to the identity. Default: 32 seconds.

Mode: Identity/Call outbound

Step	Command	Purpose
1	<code>[device](callout)# [no] non-invite-transaction-timeout <seconds></code>	Sets the transaction timeout for non-invite transactions directed to the identity. Default: 32 seconds.

Mode: Identity/Call outbound

Step	Command	Purpose
1	<code>[device](callout)# [no] penalty-box</code>	Enables or disables the use of penalty box for transactions directed to the identity. Default: disabled.

Creating an Identity Group

Multiple identities with the same properties can be grouped in an identity-group. The identity-group can be configured exactly in the same way and with the same parameters as an identity. An identity-group can only inherit configurations to identities, but they never play an active role.

The special identity-group inherits parameters to identities which are unknown or not configured. Even dynamically created identities from registration inbound inherits from the identity-group DEFAULT. Configured identities do not inherit from the identity-group DEFAULT unless is it explicitly configured to do so.

Mode: Location Service

Step	Command	Purpose
1	<code>[device](svc-ls)[ls]#(svc-ls)[ls]#[no] identity-group <name></code>	Adds a new identity-group to the location service. The no form of the command removes an existing identity-group.

Inheriting from an Identity Group to an Identity

An identity only inherits parameters from an identity-group if the parameters are not configured in the identity itself. Some commands allow the identity to explicitly disable some configurations that were otherwise inherited.

Mode: Location Service

Step	Command	Purpose
1	<code>[device](svc-ls)[ls]#identity <name> inheritance <identity-group></code>	Configures the identity to inherit parameters which are not configured from the identity-group.

Mode: Location Service

Step	Command	Purpose
1	<code>[device](svc-ls)[ls]#identity-group <name> inheritance <identity-group></code>	Configures the identity-group to inherit parameters which are not configured from another identity-group.

Mode: Location Service

Step	Command	Purpose
1	<code>[device](svc-ls)[ls]#identity <name> no-inheritance</code>	Configures the identity to not inherit parameters from the identity-group.

Mode: Location Service

Step	Command	Purpose
1	<code>[device](svc-ls)[ls]#identity-group <name> no-inheritance</code>	Configures the identity-group to not inherit parameters from the other identity-group.

Configuring the Message Waiting Indication Feature for SIP

Note Message Waiting Indication is programmed in three sections of Trinity, the FXS interface, ISDN interface, and the SIP Location service. The information below refers to information for configuring the Message Waiting Indication feature for SIP. For information on configuring the Message Waiting Indication feature for FXS, see section “[Configure Message-Waiting Indication \(MWT\) Feature for FXS](#)” on page 900. For ISDN, see section “[Configuring the Message Waiting Indication Feature for ISDN](#)” on page 821.

The SIP part of Trinity receives message waiting information according to the subscribe notify mechanism described in RFC 3265 and RFC 3842. It is possible to choose between *explicit* and *implicit subscription*. In **explicit subscription**, Trinity establishes the subscription by sending SUBSCRIBE messages to a configured message server. In **implicit subscription**, the message server gets the subscriber information due to configuration or another mechanism like registration. In any case, Trinity accepts and processes NOTIFY messages with message waiting information, which is then forwarded to the according destination.

Subscription

If an identity is added to a location-service which is bound to a context sip-gateway, the following procedure takes place:

1. Determine if identity should be subscribed.
 - The location-service containing the identity must have at least one domain configured.
 - The identity must have a message inbound face configured or inherited.
 - The subscription command must be set to “explicit”.
2. Build the address to subscribe. The name of the identity builds the user-part. The first domain configured in the location-service builds the host-part.
3. Build the address of the message server. The message server configured in the message inbound face is taken as request-uri. If no message server is configured the first domain configured in the location-service builds the request-uri.
4. Build expire header. If a lifetime is configured in the message inbound face the expire header is set to desired lifetime else the lifetime is set to 3600 seconds.
5. Build contact address. The spoofed-contact parameter of the sip-interface in the context sip gateway, through which the SUBSCRIBE request is sent, is set as contact address. If no spoofed-contact is configured the ip-address and port of the sip-interface in the context sip-gateway through which the SUBSCRIBE request is sent builds the contact address.
6. Send the SUBSCRIBE request.

If one of these steps has no result and fails, the subscription fails. After a certain timeout (which is configurable in the registration outbound face), the request is re-issued.

Outgoing SUBSCRIBE requests can provide authentication credentials.

Notification

If the gateway receives an incoming NOTIFY request, the following procedure takes place:

1. Determine to which sip interface in the context cs the request should be forwarded. This happens according to the same rules as an incoming INVITE is forwarded.
2. Get Identity. All location services bound to the context sip-gateway are searched for the identity:
 - the identity matching the to-uri
 - the identity matching request-uri
 - the identity-group default
3. Get message inbound face. The identity found must have a message inbound face configured.
4. Check subscription. The option “subscription” must be set to “implicit” or “explicit”. In the case of explicit subscription the real state of the subscription is not examined. According to RFC 3265 NOTIFY requests must be handled even before subscription is completed.
5. Check event header. The event-header of the NOTIFY request must be set to “message-summary”.
6. Check content header. If present, the content header must be set to “application/simple-message-summary”
7. Forward message waiting information. Forward content of NOTIFY message through call-control according to normal call routing to the destination provider.
8. Return 200 OK if all of these steps are successful.

If one of these steps fails the notification fails and an according message is sent.

Incoming NOTIFY requests can be challenged to provide authentication credentials.

Configuration

The message inbound face is used to subscribe an identity on a message server and enables the reception of NOTIFY messages with message waiting information.

Mode: Identity

Step	Command	Purpose
1	<code>[device](identity)[device]#[no] message inbound</code>	Adds a new face to the identity. The no form of the command removes an existing face with all content in it.

Mode: Message inbound

Step	Command	Purpose
1	<code>[device](msgin)#[no] subscribe [implicit explicit none]</code>	Enables subscription implicit, explicit or disables subscription with none.

Mode: Message inbound

Step	Command	Purpose
1	<code>[device](msgin)#[no] message server <host> [<port>]</code>	Configures the address of the message server to send your subscription requests for explicit subscription. When no message-server is configured, the subscription requests are sent to the first domain entry in the location-service.

Mode: Message inbound

Step	Command	Purpose
1	<code>[device](msgin)#[no] lifetime <seconds></code>	Configures the desired lifetime of the explicit subscription. When no lifetime is configured the desired lifetime is set to 3600 seconds.

Mode: Message inbound

Step	Command	Purpose
1	<code>[device](msgin)#[no] retry-timeout (on-client-error on-system-error on-server-error) <seconds></code>	Configures the time to wait after a failed subscription according to three different error categories. After this time we begin to subscribe from new. If no retry-timeout is configured the retry-timeout is set to 10 seconds.

Mode: Message inbound

Step	Command	Purpose
1	<code>[device](msgin)#proxy <host> [<port>] [strict-route]</code> OR <code>[device](msgin)#proxy <index> <host> [<port>] [strict-route]</code> OR <code>[device](msgin)#proxy before <index> <host> [<port>] [strict-route]</code> OR <code>[device](msgin)#proxy after <index> <host> [<port>] [strict-route]</code>	Adds a new proxy entry to the message inbound face. For each proxy configured there is a route-header added to the SUBSCRIBE requests in explicit subscription. If more than one proxy entry has to be entered, the order of the list can be modified by using the index and/or the insert keywords before and after.

Mode: Message inbound

Step	Command	Purpose
1	<code>[device](msgin)#proxy none</code>	Removes all proxy entries and disables explicitly the use of a proxy.

Message Waiting Indication through Call-Control

The message waiting indication received on the SIP side is transported through the call control to the destination interface. Currently, the SIP interface is the only source of message waiting indication information and the interface is the only possible destination of this information. The information is routed like a call through the call-control. Therefore, it must exist a valid routing path for a call from the SIP interface handling the NOTIFY to the destination interface. This must be the case, because the phone connected to the interface must be able to get calls from SIP before it makes any sense to get message waiting information for missed calls from SIP.

The message waiting information can be transported through routing-tables, service aaa, service limiter, service priority, service distribution-group and service hunt-group.

- In case of routing through **routing tables**, the message waiting information will reach the right destination only if the parameter according which the routing occurs is set to exactly the same value in an incoming INVITE request for that interface as in the incoming NOTIFY request for that interface.
- In the case of “**called-e164**”, this is surely the same, but pay attention to the manipulation of parameters.
- In case of service **distribution-group**, the message waiting information is transported to any valid destination of the distribution-group.
- In case of service **hunt-group**, the message waiting information is transported only to the first destination of the hunt group.
- In **any other of the named services**, the message waiting information is routed transparently through.

If the message waiting information is routed to a destination service or interface that is not explicitly listed here, the information will be dropped.

Show Location-Service

This command displays the location-service information of all the configured location-services and their identities. It is possible to specify the name of which location-service you want to display and also a specific identity.

Mode: Interface SIP

Step	Command	Purpose
1	<code>device>show location-service [name] [identity]</code>	Display the information of location-services and identities.

Configuration Examples

In this configuration example, inheritance is used.

Example:

```
location-service PATTON
  domain patton.com

identity-group REGISTER

  authentication outbound
    authenticate 1 authentication-service AUTH_PATTON username john

  registration outbound
    registrar sip.domain.com
    lifetime 600
    register auto

identity 300 inherits REGISTER
identity 400 inherits REGISTER
```

Exactly the same can be configured without inheritance. All inherited parameters can be configured in the identity itself. Inheritance is useful if multiple identities share the same configuration.

Example:

```
location-service PATTON
  domain patton.com

identity 300

  authentication outbound
    authenticate 1 authentication-service AUTH_PATTON username john

  registration outbound
    registrar sip.domain.com
    lifetime 600
    register auto

identity 400

  authentication outbound
    authenticate 1 authentication-service AUTH_PATTON username john

  registration outbound
    registrar sip.domain.com
    lifetime 600
    register auto
```

Chapter 49 Call Router Configuration

Chapter contents

Introduction	563
Call Router Configuration Task List	565
Map the Goals for the Call Router	565
Enable Advanced Call Routing on Circuit Interfaces	566
Configure General Call Router Behavior	566
Configure address completion timeout	566
Configure default digit collection timeout and terminating character	567
Configure Number Prefix for ISDN Number Types	568
Configure Call Routing Tables	568
Create a routing table	569
Called Party Number Routing Table	572
Regular expressions	572
Digit collection	573
Digit collection variants	574
Calling party number routing table	577
Number Type Routing Table	577
Numbering Plan Routing Table	578
Name Routing Table	579
IP Address Routing Table	579
URI Routing Table	580
Presentation Indicator Routing Table	580
Screening Indicator Routing Table	581
Information Transfer Capability Routing Table	582
Call-router Support for Redirecting Number and Redirect Reason	583
Time of Day Routing Table	583
Day of Week Routing Table	584
Date Routing Table	584
Deleting Routing Tables	584
Configure Routing Tables Dial Plan	585
Format	586
Configuration	586
Debugging	587
Status	587
Configure Mapping Tables	588
E.164 to E.164 Mapping Tables	593
Custom SIP URIs from Called-/Calling-e164 Properties	596
Other Mapping Tables	596
Deleting Mapping Tables	597
Creating Complex Functions	598

Deleting Complex Functions	599
Digit Collection & Sending-complete Behavior	600
Sending-complete	600
Ingress interface	600
Call-router	601
Egress interface	603
Creating Call Services	605
Creating a Hunt Group Service	605
Defining the Hunt-group Behavior for Inband Information	614
SIP handover on RTP loss with Hunt-Group	614
Creating a Distribution Group Service	616
Distribution-Group Min-Concurrent Setting	618
Call-router ‘limiter’ Service	618
Priority Service	619
CS Bridge Service—‘VoIP Leased Line’	620
Configuring the Service Second-dialtone	623
Deleting Call Services	624
Activate the Call Router Configuration	624
Test the Call Router Configuration	625
Configuring Partial Rerouting	631
Call reroute	631
Enable rerouting requests on ISDN.....	631
Enable emission of rerouting requests on ISDN.....	632
Enable sending of “302 moved temporary” message on SIP.....	632
Allow push-back	632
Enable push-back – aaa service	633
Enable push-back – bridge service	633
Enable push-back – distribution-group service	633
Enable push-back – hunt group service	633
Enable push-back – limiter service.....	633
Enable push-back – priority service	633

Introduction

This chapter provides an overview of call router tables, mapping tables and call services and describes the tasks involved in configuring the call router in Trinity. This chapter includes the following sections:

- Call router configuration task list
- Call router configuration tasks
- Examples

There are two options for deciding where an incoming call on a CS interface is forwarded to:

- Basic interface call routing: Basic interface routing can be configured directly on the CS interfaces. It's also called *direct call routing*.
- Advanced call routing: More complex call forwarding decisions can be configured in the call router.

The call router is a very efficient and flexible tool for routing calls between CS interfaces. Based on a set of routing criteria, the call router determines the destination (interface) for every incoming call. The forwarding decisions and features are based on a set of routing tables, mapping-tables and call services.

Each routing table is responsible for a specific routing criterion such as the called party number or the bearer capability of the call. Multiple tables can be linked together to form a decision tree. The mapping tables can be used to modify call properties like the calling and called party numbers according to the network requirements. Call services can be used in the routing path to observe the call state and spawn other calls. The hunt-group service is an example for a call service. [Figure 89](#) illustrates direct call and advanced call routing. In this chapter, advanced call routing is explained. For configuring direct call routing refer to chapter 49, "[Call Router Configuration](#)" on page 561.

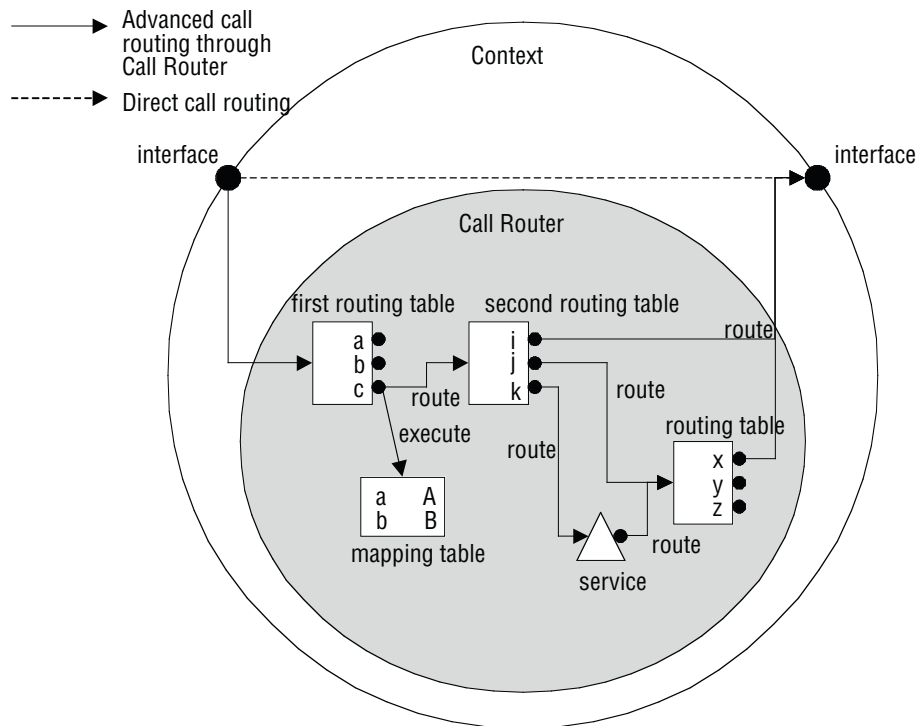


Figure 89. Direct call routing vs. advanced call routing

Due to the tree search algorithm implemented in the call router very large routing tables can be scanned very quickly with minimal impact on the call setup delay. The Trinity call router supports the following routing criteria:

- Calling party number (*calling-e164*); also called *source-Nr*, *A-Nr*, *MSN*, *DDI*, or *CLIP*
- Called party number (*called-e164*); also called *destination-Nr* or *B-Nr*
- Calling and called party number type
- Calling and called party number plan
- Calling and called party name, the display name
- Calling and called party IP address (for VoIP calls)
- Calling and called party URIs (for SIP calls)
- Presentation indicator; whether the number shall be presented to the other party
- Screening indicator; whether the number has been screened
- Information transfer capability; also called ISDN bearer capability or ISDN service
- Day of week; Monday–Sunday
- Time of day; *hour:minute:second*
- Date; *day.month.year*



The call router allows you to solve practically any call routing and call property manipulation requirement that you may have. The call router is very flexible in allowing the construction of decision trees based on linked routing tables. However you should take care not to use too many tables and an over-elaborate structure. The configuration may become large and difficult to manage. For complex configurations we recommend offline editing and configuration downloads.

Call Router Configuration Task List

To configure the call router, perform the tasks in the following sections and in the order as listed below.

- Map out the goals for the call router
- Enable advanced call routing on circuit interfaces
- Configure general call router behavior
- Configure call router tables
- Configure mapping tables and complex functions
- Configure call services
- Deleting routing tables and functions
- Activate the call router configuration
- Test the call router configuration

Map the Goals for the Call Router

There are many possible policies and factors that may influence the call router configuration. Some examples include:

- On-net off-net call routing
- ISDN service routing
- Carrier selection
- Service quality
- Fallback strategies
- Network and gateway selection

Other factors that must be taken into account include the following:

- Available number ranges (DDI, MSN, PISN)
- Potential restrictions imposed by neighboring equipment (Remote Gateways, PBXs) on the number length or range to be used.

The call router is able to accommodate almost every combination of these requirements through a customized configuration.

In order to keep this configuration compact we recommend that you first define the routing requirements and restrictions that apply to your installation. Then define the routing and mapping tables and the call services that you need to fulfill these requirements. Finally define the decision tree (i.e. the sequence in which the tables are linked together). In this step you may realize that you need multiple tables of the same type to achieve your goals. On the other hand an alternative sequence may help you to reduce the number of tables or the size of each table while still achieving the set goal. Only when you are happy with the planned tables, functions and sequence should you start configuration.

Enable Advanced Call Routing on Circuit Interfaces

To activate the advanced call routing the first routing table in the CS interface has to be specified as you can see in [figure 89](#). Make sure the route parameter on the CS interface is configured in order to forward the incoming calls to the first table of the call router.

Procedure: To enable advanced call routing on a circuit interface

Mode: Context CS

Step	Command	Purpose
1	<code>node(ctx-cs)[switch]#interface if-type if-name</code>	Change to Interface Configuration Mode to specify the entry table in the interface
2	<code>node(if-type)[if-name]#route call dest-interface interface-name</code> OR <code>node(if-type)[if-name]#route call dest-table table-name</code> OR <code>node(if-type)[if-name]#route call dest-service service-name</code>	Use these commands to route a call (1) directly to an interface specified with the <i>interface-name</i> parameter; (2) to the call router, using the <i>table-name</i> table as first routing table; (3) directly to a service specified with the <i>service-name</i> parameter.

Configure General Call Router Behavior

Configure address completion timeout

A call that is routed through a called party number routing table possibly has a called party number that is too short for a routing decision to be made. In this case the call router waits for additional digits being entered by the calling user. When the user does not enter additional digits during the address completion timeout, the call router drops the call. You can configure the address completion timeout following the procedure below. The default address completion timeout is 12 seconds and is restarted after each digit that is sent during overlap dialing.

Note The address completion timeout is active when the call router waits for mandatory digits before being able to complete call routing. Contrary to this, the digit collection timeout, described below, waits for additional optional digits.

Procedure: To configure the address completion timeout

Mode: Context CS

Step	Command	Purpose
1	<code>node(ctx-cs)[switch]# address-completion timeout <i>timeout</i></code>	Configures the address completion timeout by specifying the <i>timeout</i> in seconds. If not configured, the default address completion timeout is 12 seconds.

Example: Configure address completion timeout

```
node[switch]#address-completion timeout 20
```

Configures the address completion timeout to 20 seconds. A call with an incomplete called party number is dropped 20 seconds after receiving the last called party address update (overlap dialed digit).

Configure default digit collection timeout and terminating character

You can enter called party routing table entries that specify an incomplete number (extended by the T-indicator; refer to section “[Called Party Number Routing Table](#)” on page 572). When the call router selects such an entry, the call router waits for additional digits and starts the digit collection timeout. When the digit collection timeout elapses or when the user enters the terminating character, the call is placed to the destination specified with the routing table entry.

The digit collection timeout has a default value of 5 seconds, the terminating character is the pound character (#) by default.

Note The digit completion timeout is active when the call router waits for optional digits of a called party number before placing the call to the selected destination. Contrary to this, the address completion timeout, described above, waits for mandatory digits.

Procedure: To configure the digit collection timeout and terminating character

Mode: Context CS

Step	Command	Purpose
1	<code>node(ctx-cs)[switch]# digit-collection timeout <i>timeout</i></code>	Configures the digit collection timeout by specifying the <i>timeout</i> in seconds. If not configured, the default digit collection timeout is 5 seconds.
2	<code>node(ctx-cs)[switch]# digit-collection terminating-char <i>char</i> OR no digit-collection terminating-char</code>	Configures the digit collection terminating character by specifying the <i>char</i> . This can be any character out of 0123456789*# . The default terminating character is the pound (#). You can also use this command in the no-form to disable that the user can stop digit collection by a terminating character.

Example: Configure address completion timeout

```
node[switch]#digit-collection timeout 3
```

```
node[switch]#digit-collection terminating-char *
```

Configures the digit collection timeout to 3s. The digit-collection timeout can be stopped by the user entering the asterisk (*) character.

Configure Number Prefix for ISDN Number Types

The called and calling party numbers in an ISDN signaling message are of a defined number type; national, international or unknown. Depending on where the message originates (PSTN, mobile network, PBX) this number type may differ. Table 28 illustrates the three number types.

Table 28. ISDN number types

Type	Format Description	Example
unknown	as dialed with all leading zeros or other prefix numbers	0041 99 888xxxx
national	(area code) (local extension number)	99 888xxxx
international	(country code) (area code) (local extension number)	41 99 888xxxx

The missing prefix in the national and international number types can complicate the call router configuration, so Trinity offers the possibility to expand these numbers before entering the first call router table.

Note The configured prefix is not removed at the exit of the call router (i.e. when a destination interface is found), but the number has the number type *unknown*.

Procedure: To configure number prefix

Mode: Context CS

Step	Command	Purpose
1	<code>node(ctx-cs)[switch]#national-prefix prefix</code>	Adds <i>prefix</i> to all E.164 numbers of type national before entering the call router.
2	<code>node(ctx-cs)[switch]#international-prefix prefix</code>	Adds <i>prefix</i> to all E.164 numbers of type international before entering the call router.

Example: Configure number prefix

```
node[switch]#national-prefix 0041
```

```
Input: 99888xxxx Result: 004199888xxxx
```

```
node[switch]#international-prefix 00
```

```
Input: 4199888xxxx Result: 004199888xxxx
```

Configure Call Routing Tables

Routing tables are identified by names that can be any arbitrary string. For ease of identification the table type is typically used as part of the name.

Call router tables are created by entering the **routing-table** command, which also brings you into the routing table configuration mode. There you can add, modify or delete entries of the routing tables. Refer to the individual table types detailed below on how to configure table entries.

- Note** The sequence of the lines is not important. The call router creates a search tree out of the table lines to ensure optimal search speed.
- Note** To remove a specific entry of a table, enter the table configuration mode and use the **no**-form on a previously entered entry. To remove a whole table, use the **no** form of the **table mode** command.

Create a routing table

A routing table forwards the call to another table, interface or service based on a specific call property like the called party number or the current date. The call router provides a number of different routing table types. A routing table looks like the following:

Type	called -e164		} Header
Name	NATIONAL		
Key	Destination	Function	} Entries
001	if USVOIP-A		
001320	if USVOIP-B		
0044	if EUROVOIP		
0049	if EUROVOIP		
default	if DEFACC	ADD-PREFIX	

Figure 90. Routing table outline

Each table contains a header and one or more entries. The header declares the type of the routing table as well as its name.

The name of the routing table is unique inside the context and serves as identifier for referencing the table from other tables or interfaces. The routing table type specifies which call property the table shall examine. [Table 29](#) lists the call properties that can be used as a routing table type.

Table 29. Routing table types

Type	Description
called-e164	Route calls based on the called party E.164 number. Entries of called-e164 tables can use wildcards to summarize routes. Digit collection can be configured on a per-entry basis.
calling-e164	Route calls based on the calling party E.164 number. Entries of calling-e164 tables can use wildcards to summarize routes.

Table 29. Routing table types (Continued)

Type	Description
calling-second-e164	Route calls based on the second calling party E.164 number. For having a second calling party E.164 number available, mappings from call-properties in the incoming interface of the calls are needed. The same rules as for the normal calling party E.164 number apply.
calling-second-type-of-number	Route calls based on the second calling party number type. For having a second calling party number type available, mappings from call-properties in the incoming interface of the calls are needed. The same rules as for the normal calling party number type apply.
calling-second-numbering-plan	Route calls based on the second calling party numbering plan. For having a second calling party numbering plan available, mappings from call-properties in the incoming interface of the calls are needed. The same rules as for the normal calling party numbering plan apply.
calling-second-pi	Route calls based on the second presentation indicator. For having a second presentation indicator available, mappings from call-properties in the incoming interface of the calls are needed. The same rules as for the normal presentation indicator apply.
calling-second-si	Route calls based on the second screening indicator. For having a second screening indicator available, mappings from call-properties in the incoming interface of the calls are needed. The same rules as for the normal screening indicator apply.
called-type-of-number	Route calls based on the called party number type. ISDN distinguishes different type of numbers.
calling-type-of-number	Route calls based on the calling party number type. ISDN distinguishes different type of numbers.
called-numbering-plan	Route calls based on the called party numbering plan. ISDN distinguishes different numbering plans.
calling-numbering-plan	Route calls based on the calling party numbering plan. ISDN distinguishes different numbering plans.
called-name	Route calls based on the display name of the called party.
calling-name	Route calls based on the display name of the calling party.
called-ip	Route calls based on the signaling IP address of the destination VoIP peer.
calling-ip	Route calls based on the signaling IP address of the origination VoIP peer.
called-uri	Route calls based on the URI of the destination VoIP peer (for SIP calls: the To-URI).
calling-uri	Route calls based on the URI of the origination VoIP peer (for SIP calls: the From-URI).
calling-pi	Route calls based on the presentation indicator.
calling-si	Route calls based on the screening indicator.
itc	Route calls based on the information transfer capability (bearer capability) to distinguish speech from data calls.
time	Route calls based on the current time of day.
date	Route calls based on the current date.

Table 29. Routing table types (Continued)

Type	Description
day-of-week	Route calls based on the current day of week.
dialplan	Creates routing table based on included dial plan file and enters configuration mode

Besides the header (name and type) a routing table contains multiple entries. Each entry specifies a specific value of the routing table type and a destination interface and an optional function. When a call arrives at a routing table, the following procedure is applied:

1. Examine the call property as specified with the routing table type.
2. Select the best matching entry. This means that the key of each of the entries is compared to the call property and the entry that matches best is chosen.
3. Execute the entry. This means executing the referenced function of the entry if specified, and routing the call to the specified destination interface, table or service.

Procedure: To create a routing table and add entries

Mode: Context CS

Step	Command	Purpose
1	<code>node(ctx-cs)[switch]#routing-table table-type table-name</code>	Create a routing table <i>table-name</i> of the specified <i>table-type</i> . This enters the table mode where entries can be added or removed. To enter a previously created table from the context CS mode, you may leave away the <i>table-type</i> .
2	<code>node(rt-tab)[table-name]#route key dest-interface if-name function</code> OR <code>node(rt-tab)[table-name]#route key dest-table table-name function</code> OR <code>node(rt-tab)[table-name]#route key dest-service service-name function</code>	Add an entry to the routing table for destination interface <i>if-name</i> , destination table <i>table-name</i> or destination service <i>service-name</i> . Optionally you can specify a <i>function</i> (mapping-table or complex function) that shall be executed before the call is routed to the destination interface, table or service. The format of the <i>key</i> depends on the type of table. The next sections explain key formats for the different table types.
3		Repeat step 2 to add lines for additional table entries.

Example: Called party number routing table

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table called-e164 NATIONAL
node(rt-tab)[switch.NATIONAL]#route 001 dest-interface USVOIP-A
node(rt-tab)[switch.NATIONAL]#route 001320 dest-interface USVOIP-B
node(rt-tab)[switch.NATIONAL]#route 0044 dest-interface EUROVOIP
node(rt-tab)[switch.NATIONAL]#route 0049 dest-interface EUROVOIP
node(rt-tab)[switch.NATIONAL]#route default dest-interface DEFACC ADD-PREFIX
```

Called Party Number Routing Table

The called party number (called-e164) table is used to route calls based on the called party E.164 number in the call set-up message. The call router scans the table to find the longest matching *key* starting with the **first** digit.

Regular expressions

The *key* of an entry can be either a complete number or a partial number with wildcard digits, represented by a period (.) character. Each (.) represents a wildcard for an individual digit. For example, if the *key* is defined as 888, then any called party number beginning with 888, plus at least four additional digits matches this entry.

In addition to the period (.), there are several other symbols that can be used as wildcard characters in the *key*. These symbols provide additional flexibility in designing call routing and decrease the need for multiple entries in configuring number ranges.

The following table shows the wildcard characters that are supported:

Table 30. Wildcard symbols used as keys in E.164 tables (calling-e164, called-e164)

Symbol	Description
.	Indicates a single-digit placeholder. For example, 888 matches any dialed number beginning with 888, plus at least four additional digits. Note that the key only specifies the prefix. Thus the number may be longer, but also matches.
[]	Indicates a range of digits. A consecutive range is indicated with a hyphen (-); e.g. [5-7]. A non-consecutive range is indicated without a delimiter. For example, [58]. Both can be used in combination; e.g. [5-79], which is the same as [5679]. A (^) symbol may be placed right after the opening bracket to indicate that the specified range is an exclude list. For example, [^01] specifies the same range as [2-9]. Note: Only single-digit ranges are supported. You cannot specify the range of numbers between 99 and 102 using [99-102].
()	Indicates a pattern. For example, 888(2525). It is used in conjunction with the symbol (?), (%) or (+) or when replacing a number in a mapping table.
?	Indicates that the preceding digit or pattern occurred zero or one time. Enter Ctrl-V before entering (?) from your keyboard, since the CLI normally uses the question mark to display help texts.
%	Indicates that the preceding digit or pattern occurred zero or more times. This functions the same as the asterisk (*) used in regular expression. Here the percent (%) symbol is used to be able to handle the asterisk (*) as part of a dialed number.
+	Indicates that the preceding digit or pattern occurred one or more times.
T	Enables digit-collection for this entry. The call router pauses to collect additional dialed digits. The default digit collection timeout is 5 seconds. The collection can be aborted pressing the pound (#) key. Note: The terminating character is used only to terminate the timeout and is therefore removed from the dialed number. Note: The timeout (T) symbol is only allowed for Called Party Number table entries, while all other wildcards are also allowed for Calling Party Number tables.

The next table shows some examples of how these wildcard symbols are applied to the *key* of a table entry:

Table 31. Examples of using wildcard symbols

Expression	Description
88825.+	88825, followed by one or more wildcard digits. This expression implies that the number must contain at least 6 digits starting with 88825; for example, 888251, 8882512 or 888251234567890
88825.%	88825, followed by zero or more wildcard digits. This expression implies that the string must contain at least 88825; for example, 88825, 888256, 8882567. Note: The “%” expression postfix can be left away, because the expression is always compared as prefix to the dialed number. Thus each expression automatically contains a “%” postfix.
88825+	8882, followed by 5 repeated one ore more times; for example, 88825, 888255, or 888255555555
888(25)+	888, followed by 25 repeated one or more times; for example, 88825, 8882525 or 8882525252525
0?111	An optional 0, followed by 111; for example, 0111, 111, 11123456789
8882[56]...	8882 followed by 5 or 6, plus at least three more wildcard digits.
.%45\$	Any number that has a postfix of 45; for example, 45, 045, 0041998882545. Note: The dollar sign (\$) at the end is used to disable prefix matching.

In addition to wildcard characters, the following characters can also be used in the key of the table entry:

- Asterisk (*) and pound sign (#) – These characters can be used anywhere in the key like other digits, for example, they can be used as the leading character (e.g. *21), which is handled like normally dialed number.
- Dollar sign (\$) – Disables prefix matching. Must be used at the end of the dial string.

Digit collection

Fixed-length dialing plans, in which all the numbers have fixed length, are sufficient for most voice networks, because the telephone number strings are of known lengths. Some voice networks, however, require variable-length dial plans, particularly for international calls, which use telephone numbers of different length. Furthermore some voice networks do not support overlap dialing. In this case the call-router must collect the digits before placing a call to that network with the complete number.

If you enter the timeout T-indicator at the end of the key in a Called-Party Number table, the call router accepts a fixed-length number and then waits for additional dialed digits. The timeout character must be an uppercase T. The following example shows how the T-indicator is set to allow variable-length numbers:

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table called-e164 collect
node(rt-tab)[swtich.collect]#route 0041T dest-interface CHVoIP-A
```

In the example above, the call router accepts the digits 0041, and then waits for an unspecified number of additional digits as long as the interdigit timeout has not expired. When the interdigit timeout expires, the router places the call.

The default value for the interdigit timeout is 5 seconds and can be configured using the **digit-collection timeout** command in the context CS configuration mode. You may want to override this default timeout for a

specific entry. Just place the timeout in seconds after the T-indicator; e.g. T3 to set the inter digit timeout to 3 seconds for that entry.

The user may press the pound (#) as terminating character to immediately place the call. If the pound (#) character is entered while the router is waiting for additional digits, the pound (#) character is treated as a terminator; it is not treated as part of the dialed number sent across the network. But if the pound (#) character is entered before the router begins waiting for additional digits (meaning that the pound (#) is entered as part of the fixed-length key), then the pound (#) character is treated as a dialed digit.

For example, if the key is configured as 888 . . . T, then the entire dialed string of 888#2525 is collected, but if the dialed string is 888#252#5, the #5 at the end of the dialed number is not collected, because the final pound (#) character is treated as terminator. You can change the default terminating character using the **digit-collection terminating-char** command in the context CS configuration mode. You may want to override this default terminating character for a specific entry. Just place the character after the timeout and a comma; for example, T3,* to set the terminating character to asterisk (*).

Digit collection variants

There are three different ways how a called party routing-table can be used to perform address completion or digit collection. Consider the following examples:

```
routing-table called-e164 TAB-PREFIX
route 099 dest-interface IF-OUT
```

```
routing-table called-e164 TAB-COMPLETE
route 099.... dest-interface IF-OUT
```

```
routing-table called-e164 TAB-COLLECT
route 099T dest-interface IF-OUT
```

Now assume someone picks up a phone and dials a number using overlap dialing. After picking up the phone, an empty called party number is offered to the routing tables. All three routing tables require the called party number to contain at least the prefix 099. Thus the number is incomplete and the call router waits for additional digits being entered. Now the user presses the digit one (1). The resulting called party number is 0991. The call router is again asked for routing the call to a destination. The TAB-PREFIX table performs a prefix match with its only entry and finds out that the number is long enough, so TAB-PREFIX immediately routes the call to the destination interface IF-OUT. Unlike the first table, TAB-COMPLETE needs at least three more digits. Thus the address is not complete yet and the call router waits for more digits. TAB-COLLECT has enough digits but uses the T-indicator to perform digit-collection. The call router waits for the digit-collection timeout and then places the call to the destination interface IF-OUT.

Note There is a difference between the address completion and the digit collection timeout. The address completion timeout is active when a route is incomplete, e.g. when the dialed number of 0991 is tried to match to the entry 099.... In this case, the call router cannot forward the call to the destination unless the user enters three more digits. Thus the address completion timeout is active when the call router waits for mandatory digits. The address completion timeout can be configured using the **address-completion timeout** command in the context CS mode. If the user does not enter more digits, the address completion timeout elapses and the call is dropped. The digit collection timeout is active when a route is complete but a T-indicator

is specified on the selected route, e.g. when the dialed number of 0991 is tried to match the entry *099T*. In this case the call router waits for some period of time for the user to enter additional optional digits. The digit collection timeout can be configured using the **digit-collection timeout** command in the context CS mode. If the user does not enter more digits, the digit collection timeout elapses and the call is forwarded to the destination interface.

Example: Simple called party number routing table

The following table routes call based on the called party number. An internal number starting with 5 and containing at least 3 digits is routed to the interface that is connected to the local PBX. An international call to the US is routed to the VoIP interface USVOIP. All other calls (local, national and international) are routed to the interface that is connected to the PSTN.

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table called-e164 DIST
node(rt-tab)[switch.DIST]#route 5.. dest-interface PBX
node(rt-tab)[switch.DIST]#route 001T dest-interface
USVOIP node(rt-tab)[DIST]#route default dest-interface PSTN
```

Example: Digit collection of any number

If you want to route calls from interface A directly to interface B, wanting to collect dialed digits, you have to route calls from interface A to a routing table like the one shown in this example. This table does not require the dialed number to be of any format or length but waits for arbitrary number of digits that can be entered using overlap dialing. This allows the user to enter any number to reach the destination interface.

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table called-e164 COLLECT
node(rt-tab)[switch.COLLECT]#route T dest-interface OUT
```

Example: Called party number routing table explained

Regular expressions are very powerful for E.164 lookups. A routing table always tries to find the table entry with the best matching prefix. Determining the best match is often not a very simple process. There are several rules that define the match quality of a rule for an entered number:

1. A longer rule matches better than a shorter one.
2. An explicitly specified digit matches better than a wildcard.
3. A wildcard that include less possible digits matches better that a wildcard that include more possible digits.

Consider the following routing table:

```
routing-table called-e164 test
route 1 dest-interface IF1entry #1
route 1[0-4] dest-interface IF2entry #2
route 11 dest-interface IF3entry #3
route 111T dest-interface IF4entry #4
route default dest-interface IF5entry #5
```

Note The numbers that are normally dialed are longer than the prefixes listed in the table test. For example, if the numbering plan is defined using five digits, a user normally dials a number like 12345 to reach a destination. Anyway the lookup result must be the same for en-bloc and for overlap dialing. This example shows how the table is looked up after each overlap-dialed digit and how a destination is finally found. This selected destination is the same as if the user dialed the number en-bloc.

The following table contains a list of overlap-dialed number examples that use the routing table above for a lookup:

Dialed Number	Selected Entry	Description
empty (after picking up the phone without dialing a number beforehand)	—	The number is incomplete for entries #1-#4, which all want at least know whether the number starts with a 1. Thus no entry is selected and the call router waits for additional mandatory digits or drops the call after 12 seconds (address-completion timeout).
1	—	No entry is selected. Though the dialed number matches entry #1 there are other entries that are still incomplete (the entered number is a prefix of the entry). In this state the call router waits for additional mandatory digits or drops the call after 12 seconds (address-completion timeout).
2	#5	No entry matches, so the default entry is selected; the call is placed immediately.
11	—	No entry is selected. Though the dialed number completely matches entry #1, #2 and #3, entry #4 is still incomplete. The call router waits for additional mandatory digits or drops the call after 12 seconds (address-completion timeout).
12	#2	Entry #1 and #2 match the dialed number of 12, but entry #2 matches better because the expression is more precise (longer) than entry #1. Thus the call is immediately routed to interface IF2.
19	#1	Entry #1 is the only that matches. The call is immediately placed.
111	#4	All entries match the dialed number of 111, but entry #4 matches best because the expression is more precise (longer) than entry #1-#3. Entry #4 is selected but the call is not placed immediately because the entry contains the T-indicator. The router waits for additional digits and then places call to interface IF4 when the digit-collection timeout elapses. Note: If the user enters an additional digit during digit-collection on a T-indicator, the router must not change the destination entry anymore.
112	#3	Entry #1, #2 and #3 match the dialed number of 112. Entry #1 has only an expression of one digit while entry #2 and #3 have an expression that specify two digits. Entry #3 matches better than entry #2 because entry #3 explicitly specifies the digits while entry #2 contains a wildcard for the second digit. Thus entry #3 is selected and the call is placed immediately to interface IF3.
121	#2	Entry #1 and #2 match the dialed number of 121, but entry #2 matches better. The call is immediately placed to IF2.

Dialed Number	Selected Entry	Description
191	#1	Only entry #1 matches the dialed number of 191. Thus the call is routed immediately to interface IF1.
1111	#4	The lookup procedure is the same as for dialed number 111. The call router waits for additional digits and places the call after the digit-collection timeout to interface IF4.
1111#	#4	Same as for 1111, but the pound (#) terminates the digit collection; the call is immediately placed to interface IF4.

Calling party number routing table

The calling party number (calling-e164) table is used to route calls based on the calling-e164 in the call setup message. This number in general corresponds to the extension number of a PBX or MSN of an ISDN terminal. The table can be used to route calls from extensions, which have particular call routing requirements (i.e. Terminals which require non VoIP capable ISDN services). The call router looks for the longest match starting with the first digit of the calling party number.

Note The calling party number is sometimes inserted or modified by a PBX. Sometimes there is no calling party number at all. This all depends on the equipment you connect to the device.

Note The T-indicator cannot be used in calling party number tables. (Overlap dialing only makes sense for called party numbers).

Example: Calling party number routing table

This example shows how to create a calling party number routing table that routes calls based on the last three digits of the calling party number (the extension part). The key `.%52[35]$` means that every number that starts with any digit (.) appearing zero or more times (%) followed by 52 and a 3 or 5 matches the entry. For example, the following calling party numbers match the first entry: 0998882523 or 0998882525 or simply 523 or 525.

Note This table does not contain a default entry. All calls where the calling party number does not match to one of the entries are dropped.

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table calling-e164 EXTS
node(rt-tab)[switch.EXTS]#route .%52[35]$ dest-interface breakout
node(rt-tab)[switch.EXTS]#route .%572 dest-interface DefAcc
```

Number Type Routing Table

The calling or called party number type (calling-type-of-number or called-type-of-number) table is used to route calls based on the calling or called party type of number field in the ISDN setup message. This can be used, for example, to differentiate between national and international calls.

Note When you specified a national or international prefix using the commands `national-prefix` or `international-prefix` respectively. in the context CS configuration mode, the calling or called party number is extended with the speci-

fied prefix and the type-of-number is set to unknown in the incoming interface. Thus an international number can enter the call-router as unknown number.

Note This property is only set by incoming ISDN calls. A call that arrives the call router from a FXS or SIP interface has a number type of Unknown as those interfaces do not support the number type property.

The call router can route calls according to the following number types. These values beside default can be used for the key parameter to create a routing table entry:

- unknown—Unknown number type. This is the default value for calls that arrive through an interface that does not support the number type property.
- international—International number; the number does not include prefix or escape digits.
- national—National number; the number does not include prefix or escape digits.
- network-specific—Network specific number, used to indicate administration or service number specific to the serving network, e.g. used to access an operator.
- subscriber—Subscriber number; the number does not include prefix or escape digits.
- abbreviated—Abbreviated number.

Example: Calling type-of-number routing table

The following example routes calls with an international calling party number to the next table TAB-INCOMING-INT, calls with a national calling party number to the next table TAB-INCOMING-NAT and all other calls to the next table TAB-INCOMING-UNKNOWN.

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table calling-type-of-number TON
node(rt-tab)[TON]#route international dest-table TAB-INCOMING-INT
node(rt-tab)[TON]#route national dest-table TAB-INCOMING-NAT
node(rt-tab)[TON]#route default dest-table TAB-INCOMING-UNKNOWN
```

Numbering Plan Routing Table

The calling party numbering plan or called party numbering plan (calling-number-plan or called-numbering-plan) table is used to route calls based on the calling or called party numbering plan field in the ISDN setup message. This can be used to differentiate calls between numbers of an ISDN, data, telex, national standard or private numbering plan.

Note This call property is only set by incoming ISDN calls. A call that arrives the call router from a FXS or SIP interface has a numbering plan of unknown.

The call router can route calls according to the following numbering plans. These values beside default can be used for the key parameter to create a routing table entry:

- unknown—Unknown numbering plan. This is the default value for calls that arrive through an interface that does not support the numbering plan property.
- isdn-telephony—ISDN/Telephony numbering plan according to CCITT Recommendation E.164/E.163).

- data—Data numbering plan according to CCITT Recommendation X.121.
- telex—Telex numbering plan according to CCITT Recommendation F.69.
- national-standard—Numbering plan according to a national standard.
- private—Any private numbering plan.

Example: Calling numbering-plan routing table

The following example shows how to create a routing table with name NP that routes calls based on the calling part numbering plan. Calls with calling party numbers formed according to the ISDN/Telephony numbering plan are routed to the next table *TAB-INCOMING-ISDN*, calls with calling party numbers formed according to the data numbering plan to the next table *TAB-INCOMING-DATA* and all other calls to the next table *TAB-INCOMING-UNKNOWN*.

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table calling-numbering-plan NP
node(rt-tab)[NP]#route isdn-telephony dest-table TAB-INCOMING-ISDN
node(rt-tab)[NP]#route data dest-table TAB-INCOMING-DATA
node(rt-tab)[NP]#route default dest-table TAB-INCOMING-UNKNOWN
```

Name Routing Table

The calling display name or called display name (calling-name or called-name) table is used to route calls based on the human-readable name of the calling or called party. The key you specify in a routing table entry must be identical to the display name of the calling or called party for the entry to match.

Note Incoming SIP calls use this call property to store the display name part of the SIP URI. Other interfaces set the display name to the empty string.

Example: Calling display name routing table

This example routes calls based on the display name part of the From-URI of an incoming SIP call.

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table calling-name name
node(rt-tab)[switch.device]#route "John Smith" dest-table TAB-FROM-JOHN
node(rt-tab)[switch.device]#route Administrator dest-table TAB-FROM-ADMIN
node(rt-tab)[switch.device]#route default dest-table TAB-FROM-UNKNOWN
```

IP Address Routing Table

The calling party IP address (calling-ip) table is used to route calls based on the signaling IP address of the originating VoIP peer. The called party IP address (called-ip) table is used to route calls based on the called IP address property. The called IP address of incoming calls is set to 0.0.0.0 unless modified by a previous mapping table in the routing path. Thus the called IP address table is of limited use only.

You may specify a whole subnet with the key parameter of the routing table entry. The format of the key parameter is IPaddress[/mask-size]; the mask size may be omitted.

Note Incoming SIP calls use the calling party IP address property to store the IP address of the remote SIP user agent, respectively. Other interfaces like ISDN or FXS set the IP address to 0.0.0.0.

Example: Calling IP address routing table

The following example routes all calls from a remote SIP user agent in the LAN to the next table *TAB-FROM-LAN* and all other calls to *TAB-FROM-WAN*.

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table calling-ip IP
node(rt-tab)[switch.IP]#route 172.16.32.0/24 dest-table TAB-FROM-LAN
node(rt-tab)[switch.IP]#route default dest-table TAB-FROM-WAN
```

URI Routing Table

The calling party URI (*calling-uri*) table is used to route calls based on the URI of the originating VoIP peer, e.g. the From-URI of the incoming SIP call. The called party URI (*called-uri*) table is used to route calls based on the *To-URI*. The called URI of incoming calls is not set unless modified by a previous mapping table in the routing path.

You can use regular expressions to specify the parts of an URI that must match in order to route the call to a specified destination.

The following example shows how to create a routing table to route all SIP calls from John Smith to the next table *TAB-FROM-JOHN* while all other calls are routed to the next table *TAB-FROM-UNKNOWN*.

Mode: Context CS

Step	Command	Purpose
1	node(ctx-cs)[switch]#routing-table calling-uri name	Creates a routing-table that examines the From-URI of an incoming SIP call
2	node(rt-tab)[switch.device]#route sip:john\smith@.% dest-table TAB-FROM-JOHN	Routes all SIP calls from john.smith@<anywhere> to the table TAB-FROM-JOHN. Note that the dot (.) between john and smith must be escaped with a backslash (\), because the dot (.) means 'any character' in a regular expression.
3	node(rt-tab)[name]#route default dest-table TAB-FROM-UNKNOWN	Routes all other SIP calls to the table TAB-FROM-UNKNOWN

Presentation Indicator Routing Table

The presentation indicator (*calling-pi*) table is used to route calls based on the presentation indicator of the calling party number. A user that doesn't want its number being displays sets the presentation indicator to restricted. There is no presentation indicator on the called party number. Thus you cannot create a called-pi table.

Note Incoming ISDN calls set the presentation indicator according to the received ISDN Setup message. SIP interfaces set the presentation indicator to allowed.

The call router can route calls according to the following presentation indicators. These values beside default can be used for the key parameter to create a routing table entry:

- allowed—Presentation of the calling party number is allowed. This is the default value for calls that arrive through an interface that does not support presentation indicators.

- restricted—Presentation of the calling party number is restricted.
- interworking—The calling party number is not available due to interworking.

Note At the originating user-network interface, the presentation indicator is used for indicating the intention of the calling user for the presentation of the calling party number to the called user. You possibly want to remove the calling party number when the user set the presentation indicator to restricted. To achieve this, route restricted calls to a mapping table that sets the calling-e164 to the empty string (“”) as it is shown in the example below.

Example: Presentation indicator routing table

This example uses a pseudo routing table that just forwards all calls to the interface *IF-OUT* but first executes the mapping table *NO-CNPN*. This mapping table examines the presentation indicator and modifies the calling party number. If the presentation indicator is restricted, the calling party number is cleared. For all other presentation indicator values, the calling party number is not modified.

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table calling-pi PI
node(rt-tab)[switch.PI]#route default dest-interface IF-OUT NO-CNPN
node(rt-tab)[switch.PI]#exit
node(rt-tab)[switch]#mapping-table calling-pi to calling-e164 NO-CNPN
node(map-tab)[switch.NO-CNPN]#map restricted to ""
```

Screening Indicator Routing Table

The screening indicator (*calling-si*) table is used to route calls based on the screening indicator of the calling party number. A network validates the calling party's number and puts the validation result to the screening indicator. This allows a network to transparently pass the calling party number set by the calling user, but tell the called user whether or not the number selected by the calling party really belongs to him or her.

Note Incoming ISDN calls set the screening indicator according to the received ISDN Setup message. Other interfaces set the screening indicator to user-not-screened.

The call router can route calls according to the following screening indicators. These values beside default can be used for the key parameter to create a routing table entry:

- user-not-screened—The calling party number is provided by the user but not screened by the network. Thus the calling party possibly send a number that is not owned by the calling party. (This is the default value for calls that arrive through an interface that does not support presentation indicators).
- user-passed—The calling party number is provided by the user, screened by the network and really belongs to the calling party.
- user-failed—The calling party number is provided by the user, screened by the network and does not belong to the calling party.
- network—The calling party number, because it is provided by the network, can be trusted.

Note You possibly want to remove the calling party number when the calling party number is not screened or screening failed. To achieve this, route these calls

to a mapping table that sets the calling-e164 to the empty string (“”). If you want to drop calls when the calling party number is not screened or screening failed, use the routing destination none.

Example: Screening indicator routing table

A call that is routed to this table is examined for the screening indicator of the calling party number. If the calling party provided a number that was not accepted by the network (user-failed), we drop the call. Else we route the call to the interface *IF-OUT* but first execute the mapping table *NO-CNPN*. This mapping table again examines the screening indicator. If the user provided a number that was not screened by the network, the table sets the calling party number to an empty string. For all other screening indicator values, the calling party is not modified.

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table calling-si SI
node(rt-tab)[switch.SI]#route user-failed none
node(rt-tab)[switch.SI]#route default dest-interface IF-OUT NO-CNPN
node(rt-tab)[switch.SI]#exit
node(rt-tab)[switch]#mapping-table calling-si to calling-e164 NO-CNPN
node(map-tab)[switch.NO-CNPN]#map user-not-screened to ""
```

Information Transfer Capability Routing Table

The information transfer capability (*itc*) table is used to route calls based on the information transfer capability field of the bearer capability information element in the ISDN setup message. This can be used to differentiate between ISDN data services and ISDN speech connections.

Note Terminals connected to analog extensions (e.g. of a PBX) do not supply information transfer capability values in their call set-up, so it is up to the configuration of the analog port on the Terminal Adapter, NT or PBX to insert this value. The configuration of this value is however often omitted or wrong. The ITC value may therefore not be a reliable indication to differentiate between analogue speech, audio or Fax Group 3 connections. Also, calls from SIP and FXS interfaces do not differentiate between bearer capabilities. They always set the information transfer capability property to *3.1kHz Audio*.

The call router can route calls according to the following information transfer capabilities. These values beside default can be used for the key parameter to create a routing table entry:

- speech—Voice terminals (Telephones)
- unrestricted-digital—Unrestricted digital information (64kBit/s)
- restricted-digital—Restricted digital information (64kBit/s)
- 3k1-audio—Transparent 3.1kHz audio channel. This is the default value set by interfaces that do not support the ITC property.
- 7k-audio—Transparent 7.1kHz audio channel
- video—Video conference terminals

Example: Information transfer capability routing table

The following example creates an itc routing table that routes all unrestricted digital calls to the interface *IF-ACCESS*, all 7kHz audio calls to the interface *IF-LOCAL-BREAKOUT*, all video calls to the interface *IF-ACCESS* and all other calls to the interface *IF-VOIP-CARRIER-A*.

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table itc ITC
node(rt-tab)[ITC]#route unrestricted-digital dest-interface IF-ACCESS
node(rt-tab)[ITC]#route 7k-audio dest-interface IF-LOCAL-BREAKOUT
node(rt-tab)[ITC]#route video dest-interface IF-ACCESS
node(rt-tab)[ITC]#route default dest-interface IF-VOIP-CARRIER-A
```

Call-router Support for Redirecting Number and Redirect Reason

The call.router can be used to manipulate and make routing decisions depending on the call-control redirecting number and redirect reason properties. The following command creates a call-router routing-table, which uses the redirecting number for routing decisions. The contents of the routing table are the same as for any other E.164 number based call-router routing table.

Mode: context cs

Step	Command	Purpose
1	[name] (ctx-cs)[router]# routing-table calling-redir-e164 <table-name>	Creates a redirecting number routing table.

The following command creates a redirect-reason call-router routing-table. Possible reason values for routing decisions are:

- cd: Call deflection
- cfb: Call forwarding on busy
- cfd: Call forwarding by called DTE
- cfnr: Call forwarding on no reply
- cfu: Call forwarding unconditional
- ooo: Called DTE out of order
- default: Any other unhandled case

Mode: context cs

Step	Command	Purpose
1	[name] (ctx-cs)[router]# routing-table calling-redir-reason <table-name>	Creates a redirect reason routing table.

Both the redirecting-number and the redirect-reason can also be used in any call-router mapping tables.

Time of Day Routing Table

The time table is used to route calls based upon the current system time during one day, i.e. an 24hr. period from midnight to midnight. Times are matched within the ranges defined in the time routing table.

The key parameter of the routing table entry has the format: *hh:mm:ss-hh:mm:ss*

The full range must be specified. The range must not cross a day boundary at midnight.

Example: Time of day routing table

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table time WORKDAY
node(rt-tab)[switch.WORKDAY]#route 08:00:00-16:59:59 dest-table TAB-BEST-QUAL
node(rt-tab)[switch.WORKDAY]#route 17:00:00-20:59:59 dest-interface IF-VOIP-A
node(rt-tab)[switch.WORKDAY]#route 21:00:00-23:59:59 dest-interface IF-VOIP-B
node(rt-tab)[switch.WORKDAY]#route 00:00:00-07:59:59 dest-interface IF-VOIP-B
```

Day of Week Routing Table

The day-of-week table is used to route calls according to the day of the week. The days are defined by the long lowercase names Monday; Tuesday; Wednesday; Thursday; Friday; Saturday; and Sunday. To configure week-day routing table entries use the following commands starting in the CS context configuration mode.

Example: Day of week routing table

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table day-of-week TAB-DAY
node(rt-tab)[switch.TAB-DAY]#route saturday dest-table TAB-LEAST-COST
node(rt-tab)[switch.TAB-DAY]#route sunday dest-table TAB-LEAST-COST
node(rt-tab)[switch.TAB-DAY]#route default dest-interface IF-VOIP
```

Date Routing Table

The date table is used to route calls according to the current system date. It can be used, for example, to represent holidays in the routing decision tree. The table matches exact dates or date ranges.

The key parameter of the routing table entry has the format: *dd:mm:yyyy-dd:mm:yyyy*

The full range must be specified.

Example: Date routing table

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table day-of-week HOLIDAY2001
node(rt-tab)[switch.HOLIDAY~]#route 01.01.2001-02.01.2001 dest-table TAB-HOL
node(rt-tab)[switch.HOLIDAY~]#route 05.01.2001-05.01.2001 dest-table TAB-HOL
node(rt-tab)[switch.HOLIDAY~]#route 24.12.2001-31.12.2001 dest-table TAB-HOL
node(rt-tab)[switch.HOLIDAY~]#route default dest-interface IF-VOIP
```

Deleting Routing Tables

To remove individual routing tables you can use the **no** form of the **routing table** command. Alternatively you can remove specific entries of a routing table by entering the routing table configuration mode and use the **no** form of the **route** command.

Procedure: To delete an entry from a routing table

Mode: Context CS

Step	Command	Purpose
1	<code>node(ctx-cs)[switch]#routing-table table-name</code>	Enter the routing table from which you want to remove an entry. Note: You do not have to enter the type of the table when just entering it. The type must only be specified when creating a table.
2	<code>node(rt-tab)[table-name]#no route key</code>	Remove the entry with the specified <i>key</i> .
3		Repeat Step 2 to remove additional entries.

Example: Remove entries from a routing table

The running-config shows the following table:

```
routing-table called-e164 MY-TABLE
  route 10 dest-interface IF1
  route 11 dest-interface IF2
  route 12 dest-interface IF3
  route default dest-interface IF4
```

To remove the first two entries from the table enter the following commands:

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table MY-TABLE
node(rt-tab)[switch.MY-TABLE]#no route 10
node(rt-tab)[switch.MY-TABLE]#no route 11
```

The resulting running-config is:

```
routing-table called-e164 MY-TABLE
  route 12 dest-interface IF3
  route default dest-interface IF4
```

Procedure: To delete an entire routing table

Mode: Context CS

Step	Command	Purpose
1	<code>node(ctx-cs)[switch]#no routing-table table-name</code>	Delete the routing table <i>table-name</i> . Note: You do not have to enter the type of the table when just deleting it. The type must only be specified when creating a table.

Example: Remove an entire routing table

```
node(cfg)#context cs
node(ctx-cs)[switch]#no routing-table MY-TABLE
```

Configure Routing Tables Dial Plan

A routing table dial plan is very similar to a called-e164 routing table. The main differences between a dial plan and routing table are the reduced pattern syntax, and the fact that the dial plan is stored in a text file in the

Trinity's flash memory. Additionally, it is possible to provide the dial plan file with the Trinity's provisioning feature.

Format

A dial plan file contains one or more lines of entries that are formatted like:

PATTERN;DESTINATION-TYPE;DESTINATION-NAME

PATTERN: is divided by a comma in a prefix and a length. The length specifies the total length of a phone number that has to be dialed to match the prefix. If the length is 0, no more digits are expected after the prefix. If the length is less or equal to the prefix length, then it's assumed you meant 0.

For example, the PATTERN: **12345,7** would match this number: **1234511**

Three types of prefixes are possible:

- Plain number: 0123
- Number with range: 012[3-7] or 012[3-6][5-9]
- Big range: 1234-1300

Note Big ranges are not allowed to contain ranges themselves.

DESTINATION-TYPE and **DESTINATION-NAME:** are not mandatory. If the two fields are missing, the call will be routed to the default route or to the dialplan-match route. Entries without a specific destination cannot be routed if the default route or dialplan-match route entries are missing.

- **DESTINATION-TYPE:** can be an interface, service or table.
- **DESTINATION-NAME:** is the configured name of the interface, service or table.

Some valid examples of dial plan entries include:

Command	Purpose
114,5;interface;IF_BRI	114xx numbers match. Routed to call-control interface IF_BRI
115,3;service;SRV_HUNTGROUP	115 only match. Routed the call-control service SRV_HUNTGROUP
11601-11610,5;table;TBL_EXTERNAL	All numbers from 11601 to 11610 will match. Routed to call-control routing-table TBL_EXTERNAL
2515[5-9],9;	Numbers from 25155xxxx to 25159xxxx will match. Routed to dialplan-match route. If no dialplan-match route is configured, the call will be dropped.

Configuration

This section contains all commands for specifying the routing-table dial plan configuration.

Example

```
context cs SWITCH
```

```

routing-table dialplan called-e164 TBL_DIALPLAN
include dialplan:tbl-dialplan
route default dest-interface IF_ISDN_00
route dialplan-match dest-interface IF_SIP

```

Mode: Context CS

Step	Command	Purpose
1	[node] (ctx-cs)[SWITCH]#[no] routing-table dialplan called-e164 <table-name>	Creates or removes a routing-table dialplan.
2	[node](rt-tab-d)[NAME]#include dialplan:<file-name>	Uses routing entries from a downloaded dialplan file.
3	[node](rt-tab-d)[NAME]# route default ...	Sets the default route where to route calls where that don't match any entry. See routing-table configuration for details.
4	[node](rt-tab-d)[NAME]# route dialplan-match ...	Sets the dialplan-match route where to route calls where that match dialplan entry. See routing-table configuration for details.

To download a dial plan file, you can use Trinity's copy command:

```
copy tftp://<server>/<dialplan-file-name> dialplan:tbl-dialplan
```

You can use Trinity's provisioning feature to provide periodically a dial plan file on your devices (see section “Provisioning Profile” on page 186).

Debugging

When configuring a routing table dial plan it's a good idea to turn on the call-router debug:

```
debug cr
```

The log output will tell you if your dial plan contains invalid entries and on which line you can find them.

```

CR # [SWITCH] Reloading tables now
CR # [SWITCH] Flushing all tables
CR # [TBL_DIALPLAN] (23): Ambiguous(maybe duplicated) pattern: 116,3;
CR # [TBL_DIALPLAN] (65): Pattern contains invalid character: 1f3d,4;
CR # [TBL_DIALPLAN]: Added 121 entries, (Warnings: 0). Skipped: 2 erroneous
entries.

```

Status

If you don't want to enable debug output you can also check the call-routers state with the following command that will print a report on the last call-router reload:

```

# show call-router SWITCH reload
[TBL_DIALPLAN] (2): Ambiguous(maybe duplicated) pattern: 116,3;
[TBL_DIALPLAN] (3): Pattern contains invalid character: 1f3d,4;
IF_ISDN_00-precallservice: OK
IF_SIP-precallservice: OK

```

To get information on a specific routing table dial plan, runs the show command:

```
# show call-router status TBL_DIALPLAN
Lookup Table: TBL_DIALPLAN
=====

Condition Type:                called-
Modifier Type:                 called-
Dial Plan File:                dialplan:tbl-dialplan
Invalid entries:               2
Routing entries:               121
  (Warnings):                  0
Invalid entries:               2
Tree Nodes:                    124
Tree Nodes Limit:              20000

Default Element
-----

Next Table #1:                  IF_ISDN_00-dest

Dialplan-match Element
-----

Condition:                     Timeout Prefix Expression = ^(?:dialplan-
match) (timeout=none, termination=#)
Next Table #1:                  IF_SIP-dest
```

Configure Mapping Tables

Mapping tables are used to modify the call setup message and thus influence the routing decision and or the outgoing setup message leaving the call router. Mapping tables are identified by a name (string) and referenced in the routing tables for execution. Like the routing table, a mapping table finds the best matching entry and executes it; but unlike the routing table, the execution means manipulating a call property. Thus a mapping table always examines one call property (the input-type) and changes another property (the output-type). As for the routing tables, the call router provides a number of different mapping table types. A mapping table looks like the following:

Input -Type	calling -pi	} Heade
Output -Type	calling- e164	
Name	REMOVE-CNPN	
Key	Value	} Entries
restricted	if USVOIP-A	

Figure 91. Mapping table outline

Each table contains a header and one or more entries. The header declares the input and output-type of the mapping table as well as its name.

Unlike a routing table, a call property pair characterizes a mapping table, the input and output-type. While the input-type defines which call property is examined by the call router, the output-type defines which property is

modified once the best matching entry is found, for example, you may want to find a best matching entry in a mapping table based on the presentation indicator and, once found, you want to manipulate the calling party number of the call. In this case you chose an input-type of calling-pi and an output-type of calling-e164

There are three different kinds of call properties, calling party properties, called party properties and generic properties. When a call setup is to be routed by the call router, most properties appear as calling and as called party properties, for example, you have a calling party number and a called party number to examine. There are other properties (e.g. the information transfer capability), which is a property of the whole call and not of either party.

You can create a mapping table that examines and modifies a specific kind of property, e.g. the called party number. In this case you have to specify an input-type of called-e164 and an output-type of called-164. If you want to replace both, the called and the calling party property with the same mapping table, you can create a mapping table with input-type e164 and output-type e164, i.e. without prefixing the input- and output-type with “called-”.

The name of the mapping table is unique inside the context and serves as identifier for referencing the mapping table from other routing tables. Almost every type explained with the routing table above can be used as input and output-type of a mapping table.

Table 32. Mapping table types

Type	Description Input-Type	Description Output-Type
called-e164	Select an entry based on the called party E.164 number. You can use wildcards to summarize entries.	Modifies the called party E.164 number. If the input-type is a called-e164 or a calling-e164 type, you can use replacement operators to use parts of the lookup key.
calling-e164	Select an entry based on the calling party E.164 number. You can use wildcards to summarize entries.	Modifies the calling party E.164 number. If the input-type is a called-e164 or a calling-e164 type, you can use replacement operators to use parts of the lookup key.
e164	If the output-type is also a generic kind of property, this mapping table is applied to both, this calling-e164 and the called-e164 property.	If the input-type is also a generic kind of property, this mapping table is applied to both, the calling-e164 and the called-e164 property.
called-type-of-number	Select an entry based on the called party number type. ISDN distinguishes different type of numbers.	Sets the called party number type.
calling-type-of-number	Selects an entry based on the calling party number type. ISDN distinguishes different type of numbers.	Sets the calling party number type.
type-of-number	If the output-type is also a generic kind of property, this mapping table is applied to both, this calling-type-of-number and the called-type-of-number.	If the input-type is also a generic kind of property, this mapping table is applied to both, the calling-type-of-number and the called-type-of-number property.

Table 32. Mapping table types (Continued)

Type	Description Input-Type	Description Output-Type
called-numbering-plan	Selects an entry based on the called party numbering plan. ISDN distinguishes different numbering plans.	Sets the called party numbering plan.
calling-numbering-plan	Selects an entry based on the calling party numbering plan. ISDN distinguishes different numbering plans.	Sets the calling party numbering plan.
numbering-plan	If the output-type is also a generic kind of property, this mapping table is applied to both, this calling-numbering-plan and the called-numbering-plan.	If the input-type is also a generic kind of property, this mapping table is applied to both, the calling-numbering-plan and the called-numbering-plan property.
called-name	Selects an entry based on the display name of the called party.	Sets the display name of the called party.
calling-name	Selects an entry based on the display name of the calling party.	Sets the display name of the calling party.
name	If the output-type is also a generic kind of property, this mapping table is applied to both, this calling-name and the called-name.	If the input-type is also a generic kind of property, this mapping table is applied to both, the calling-name and the called-name property.
called-ip	Selects an entry based on the remote signaling IP address of the destination VoIP peer.	Sets the remote IP address of the destination VoIP peer.
calling-ip	Selects an entry based on the remote signaling IP address of the origination VoIP peer.	Sets the remote IP address of the origination VoIP peer.
ip	If the output-type is also a generic kind of property, this mapping table is applied to both, this calling-ip and the called-ip.	If the input-type is also a generic kind of property, this mapping table is applied to both, the calling-ip and the called-ip property.
calling-pi	Selects an entry based on the presentation indicator.	Sets the presentation indicator.
calling-si	Selects an entry based on the screening indicator.	Sets the screening indicator.
calling-second-e164	Select an entry based on the second calling party E.164 number. For having a second calling party E.164 number available, mappings from call-properties in the incoming interface of the calls are needed. The same rules as for the normal calling party E.164 number apply.	Modifies the second calling party E.164 number. The same rules as for the normal calling party E.164 number apply.

Table 32. Mapping table types (Continued)

Type	Description Input-Type	Description Output-Type
calling-second-type-of-number	Select an entry based on the second calling party number type. For having a second calling party number type available, mappings from call-properties in the incoming interface of the calls are needed. The same rules as for the normal calling party number type apply.	Modifies the second calling party number type. The same rules as for the normal calling party number type apply.
calling-second-numbering-plan	Select an entry based on the second calling party numbering plan. For having a second calling party numbering plan available, mappings from call-properties in the incoming interface of the calls are needed. The same rules as for the normal calling party numbering plan apply.	Modifies the second calling party numbering plan. The same rules as for the normal calling party numbering plan apply.
calling-second-pi	Select an entry based on the second presentation indicator. For having a second presentation indicator available, mappings from call-properties in the incoming interface of the calls are needed. The same rules as for the normal presentation indicator apply.	Modifies the second presentation indicator. The same rules as for the normal second presentation indicator apply.
calling-second-si	Select an entry based on the second screening indicator. For having a second screening indicator available, mappings from call-properties in the incoming interface of the calls are needed. The same rules as for the normal screening indicator apply.	Modifies the second screening indicator. The same rules as for the normal second screening indicator apply.
itc	Selects an entry based on the information transfer capability (bearer capability) to distinguish speech from data calls.	Sets the information transfer capability.
time	Route calls based on the current time of day.	Cannot be set.
date	Route calls based on the current date.	Cannot be set.
day-of-week	Route calls based on the current day of week.	Cannot be set.

Besides the header (name, input and output-type) a mapping table contains multiple entries. Each entry specifies a specific value of the routing table input-type and a value that shall be applied to the call property specified with the output-type of the table. When a call arrives a mapping table, the following procedure is applied:

1. Examine the call property as specified with the mapping table input-type.
2. Select the best matching entry. This means that the key of each of the entries is compared to the call property and the entry that matches best is chosen.

- Execute the entry. This means replacing the property specified with the output-type of the mapping table with the value of the selected entry.

Let's examine the mechanism of mapping tables in more detail. Figure 92 shows three mapping tables and a call that is routed through this mapping table. The call contains various call properties that are examined and modified by the mapping tables:

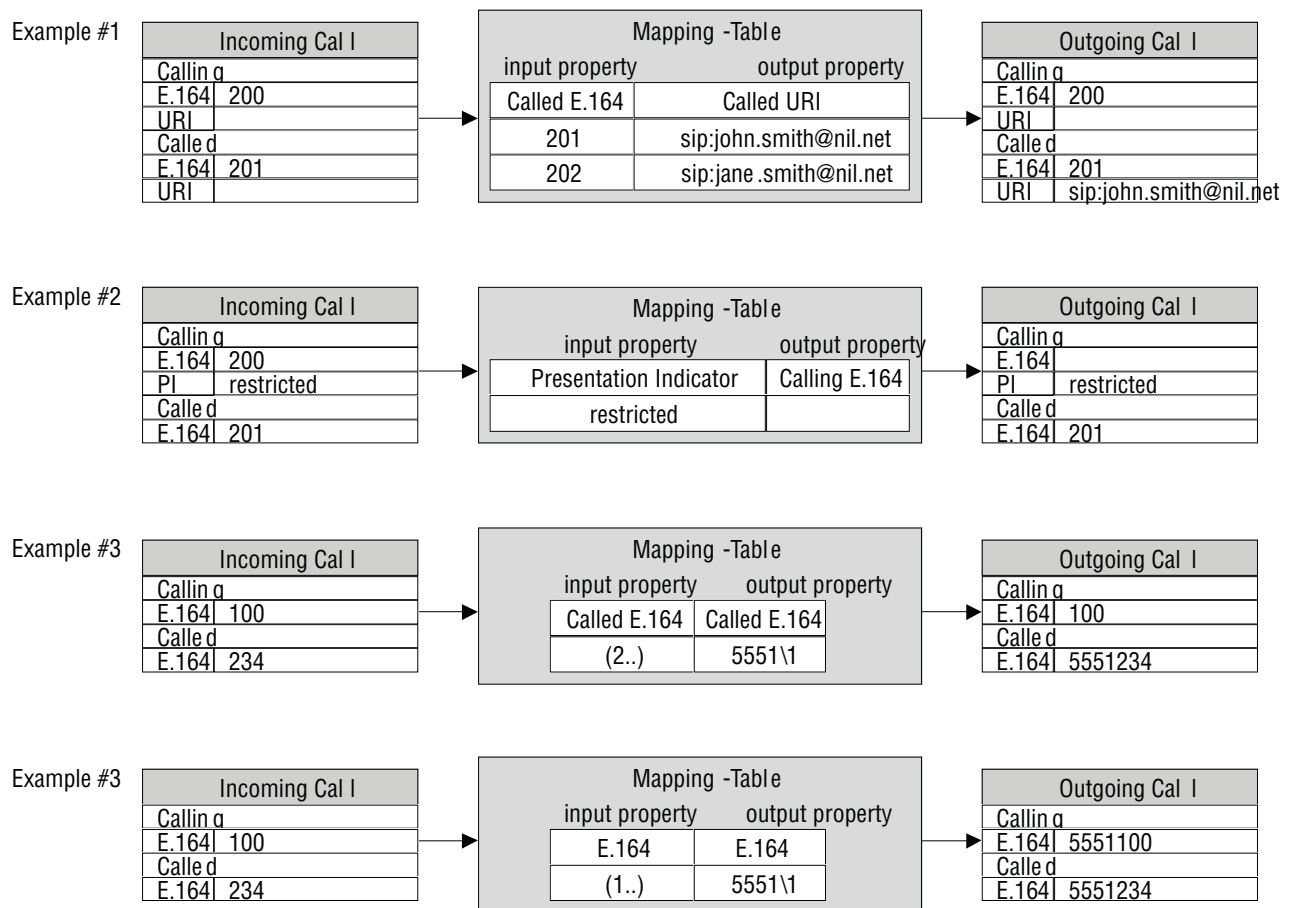


Figure 92. Mapping table examples

Example #1: This example shows a mapping table that selects the best matching entry based on the called party number of a call and, once found, sets the called party URI. In the example a call arrives to the mapping table with a called party number of 201. The mapping table selects the first entry, which matches best, and sets the called party URI of the call to *sip:john.smith@nil.net*.

Example #2: This example shows a mapping table that selects the best matching entry based on the presentation indicator and, once found, sets the called party number. In the example a call arrives to the mapping table with a presentation indicator of *restricted*. The mapping table selects the only entry (which matches) and sets the called party number to the empty string.

Example #3: This example shows a mapping table that selects the best matching entry based on the called party number and, once found, changes the same property, the called party number. This is a very powerful

method to manipulate numbers using regular expressions. In this example a call arrives to the mapping table with a called party number of 234. The mapping table selects the only entry (which matches) and adds a prefix of 5551 to the called party number.

Example #4: This example shows the same input call properties as in example #3. The mapping table is also almost the same, but unlike in the previous example, here we don't look for a specific number type (e.g. called party number, calling party number), but for any E.164 number property of the call. The output property is also a generic number. In this case the mapping table replaces both, the calling and the called party number. For example, the mapping table applies its algorithm to all E.164 numbers of the call.

Note Like a routing table, a mapping table selects the best matching entry based on the input property type. This is done using the best matching prefix method for E.164 numbers and string compare for other properties. Unless you don't have a default entry in a mapping table, no action is performed when no match can be found and the call is not dropped. In the above example #3, if a call arrives the mapping table with a called party number of 200, which does not match the entry (1..), the called party number is not changed.

Procedure: To create a mapping table and add entries

Mode: Context CS

Step	Command	Purpose
1	<code>node(ctx-cs)[switch]#mapping-table <i>input-type</i> to <i>output-type</i> <i>table-name</i></code>	Create a mapping table <i>table-name</i> that examines the call property specified with <i>input-type</i> and modifies the call property specified with <i>output-type</i> . This enters the table mode where entries can be added or removed. To enter a previously created table from the context CS mode, you may leave away the <i>input-type</i> and <i>output-type</i> .
2	<code>node(map-tab)[table-name]#map <i>key</i> to <i>value</i></code>	Add an entry to the mapping that sets the <i>output-type</i> call property to <i>value</i> if the <i>output-type</i> call property matches the <i>key</i> . The format of the <i>key</i> depends on the <i>input-type</i> of the table, and the format of the <i>value</i> depends on the <i>output-type</i> of the table.
3		Repeat step 2 to add lines for additional table entries.

Example: Called and calling party manipulation mapping table

```
node(cfg)#context cs
node(ctx-cs)[switch]#mapping-table e164 to uri SETURI
node(rt-tab)[switch.SETURI]#map 100 to sip:john.smith@nil.net
node(rt-tab)[switch.SETURI]#map 101 to sip:jane.smith@nil.net
```

E.164 to E.164 Mapping Tables

As with routing tables you can use regular expressions when selecting an entry in a mapping table based on a calling or called party number. If the output property type of a mapping table is also a calling or called party number, you may use parts of the matched expressions when building the modified number as shown in example #3 above.

Detailed Example: You have an internal dial plan that uses three digit numbers starting with a 2 (e.g. 200, 201, etc.). So when an internal subscriber makes a call, its calling party number contains three digits.

1. You want to route calls to the public switched telephone network (PSTN), that is reachable over and ISDN interface. From the PSTN provider you have an assigned number range from 099-8882500 to 099-8882599.
2. You want to pass the last two digits of your internal subscribers when they are making calls to the PSTN. Thus subscriber 244 should make a call to the PSTN using a calling party number of 099-8882544.

To achieve this, create a mapping table that looks like the following:

```
node(cfg)#context cs
node(ctx-cs)[switch]#mapping-table calling-e164 to calling-e164 MAP-PSTN
node(rt-tab)[switch.MAP-PSTN]#map 2(..) to 09988825\1
```

When a call reaches this table with a calling party number of 244, this number is tried to match to the entries of this table:

```
2(44) matches 2(. .)
```

Thus the only entry is selected and executed. This means setting the calling party number to 09988825\1. The last part of the value (a backslash followed by a single digit number) is a placeholder and means that the first pattern (expression in brackets) of the key shall be used instead.

Thus the called party number is replaced with the specified prefix 09988825 concatenated with the bracketed pattern in the key (44). The result is 0998882544.

Like this you can use brackets around any party of the expression of the key and use the part that matches to this bracket in the value you set.

Example: Mapping table to add a prefix to the called party number

Input:called-e164 = 0998882525

Output:called-e164 = *50998882525

```
node(cfg)#context cs
node(ctx-cs)[switch]#mapping-table called-e164 to called-e164 ADD-PFX
node(rt-tab)[switch.ADD-PFX]#map (.%) to *5\1
```

The input 0998882525 matches the expression (.%)
– any character repeated zero or more times.

The first bracket encloses the whole number: (.%)
== (0998882525) -> \1 = 0998882525

The output is built as concatenation of *5 and the first bracket \1.

The called party number is set to *50998882525

Example: Mapping table to remove a prefix from the called party number

Input:called-e164 = *50998882525

Output:called-e164 = 0998882525

```
node(cfg)#context cs
node(ctx-cs)[switch]#mapping-table called-e164 to called-e164 REM-PFX
node(rt-tab)[switch.REM-PFX]#map *5(.%) to \1
```

The input *50998882525 matches the expression *5(.%) – the prefix *5 followed by any character repeated zero or more times.

The first bracket encloses the number after the prefix: *5(.%) == *5(0998882525) -> \1 = 0998882525

The output is built from the first bracket \1.

The called party number is set to 0998882525.

Example: Mapping table to truncate the called party number

Input:called-e164 = 0998882525

Output:called-e164 = 525

```
node(cfg)#context cs
node(ctx-cs)[switch]#mapping-table called-e164 to called-e164 TRUNC
node(rt-tab)[TRUNC]#map .%(...) to \1
```

The input 0998882525 matches the expression .%(...) – any character repeated zero or more times followed by three mandatory digits.

The first bracket encloses the last three digits: .%(...) == 0998882(525) -> \1 = 525

The output is built from the first bracket \1.

The called party number is set to 525.

Example: Mapping table to remove the calling party number when restricted

Input:calling-e164 = 0998882525; calling-pi = restricted

Output:calling-e164 = ""; calling-pi = restricted

```
node(cfg)#context cs
node(ctx-cs)[switch]#mapping-table calling-pi to calling-e164 REM-CNPN
node(rt-tab)[switch.REM-CNPN]#map restricted to ""
```

The input (presentation indicator) restricted matches the expression restricted.

The output (calling party number) is an empty string ("").

The calling party number is cleared.

Output:calling-e164 = 0778881111; called-e164 = 0778881111

```
node(cfg)#context cs
node(ctx-cs)[switch]#mapping-table called-e164 to calling-e164 COPY-PN
node(rt-tab)[switch.COPY-PN]#map (.%) to \1
```

The input called party number 0778881111 matches the expression (.%) – any character repeated zero or more times.

The first bracket encloses the last whole called party number: (.%) == (0778881111) -> \1 = 0778881111

The output (calling party number) is built from the first bracket \1.

The calling party number is set to 0778881111.

Custom SIP URIs from Called-/Calling-e164 Properties

The regular expression engine used for mapping-tables and routing-tables in the Call Router can handle not only handle digits, but whole strings. You can construct custom SIP URIs from call leg properties as called and calling e.164. See the following examples.

Example 1: Use regular expressions to create mapping tables that map the called-party-e164 number to the called-party-URI for SIP calls. The following example shows how to build a SIP To-URI from the called-party number.

Mode: Context CS

Step	Command	Purpose
1	<code>node(ctx-cs)[switch]#mapping-table called-e164 to called-uri name</code>	Creates a mapping table that examines the called-party number and sets the called-party URI (To-URI).
2	<code>node(map-tab)[switch.device]#map (.+) to sip:user_\1@example.com</code>	If the called-party number exists (at least one digit) the called-party URI is set to user_<called-e164>@example.com
3	<code>node(map-tab)[name]#map default to sip:anonymous@example.com</code>	If the called-party number does not exist (calls that don't match to the rule of step 2), the called-party URI is set to anonymous@example.com

Example 2: Use a mapping table to set the display name field of To-URIs for outgoing SIP calls from the called-party number of an incoming call. The following example shows how to set the called-party name based on the called-party number.

Mode: Context CS

Step	Command	Purpose
1	<code>node(ctx-cs)[switch]#mapping-table called-e164 to called-name name</code>	Creates a mapping table that examines the called-party number and sets the called-party name.
2	<code>node(map-tab)[switch.device]#map (.%) to \1</code>	Copies the whole called-party number to the called-party name.

Other Mapping Tables

Example: Mapping table to set the called party number type to international (unconditionally)

Input:called-e164 = 0041998882525; calling-type-of-number = unknown

Result:called-e164 = 0041998882525; calling-type-of-number = international

```
node(cfg)#context cs
node(ctx-cs)[switch]#mapping-table called-type-of-number to called-type-of-number
SET-INT
node(rt-tab)[SET-INT]#map default to international
```

Any called party number type matches the default entry. Note that the input-type of the table does not matter when the mapping table contains only the default entry. Anyway an input-type must be specified when creating the mapping table.

The output (called party number type) is international.

The called party number type is set to international.

Example: Mapping table to replace called party numbers (translation table)

```
node(cfg)#context cs
node(ctx-cs)[switch]#mapping-table called-e164 to called-e164 TRANS
node(rt-tab)[switch.TRANS]#map 550 to 250
node(rt-tab)[switch.TRANS]#map 551 to 251
node(rt-tab)[switch.TRANS]#map 552 to 252
node(rt-tab)[switch.TRANS]#map 553 to 253
node(rt-tab)[switch.TRANS]#map 554 to 254
node(rt-tab)[switch.TRANS]#map 555 to 255
node(rt-tab)[switch.TRANS]#map 556 to 256
node(rt-tab)[switch.TRANS]#map 557 to 257
node(rt-tab)[switch.TRANS]#map 558 to 258
node(rt-tab)[switch.TRANS]#map 559 to 259
```

Note The translation table above can be reduced using regular expressions.

```
node(cfg)#context cs
node(ctx-cs)[switch]#mapping-table called-e164 to called-e164 TRANS
node(rt-tab)[switch.TRANS]#map 55(.) to 25\1
```

Deleting Mapping Tables

To remove individual mapping tables you can use the **no** form of the **mapping table** command. Alternatively you can remove specific entries of a mapping table by entering the mapping table configuration mode and use the **no** form of the **map** command.

Procedure: To delete an entry from a mapping table

Mode: Context CS

Step	Command	Purpose
1	<code>node(ctx-cs)[switch]#mapping-table table-name</code>	Enter the mapping table from which you want to remove an entry. Note: You do not have to enter the type of the table when just entering it. The type must only be specified when creating a table.
2	<code>node(map-tab)[table-name]#no map key</code>	Remove the entry with the specified key.
3		Repeat Step 2 to remove additional entries.

Example: Remove entries from a mapping table

The running-config shows the following table:

```
mapping-table called-e164 to called-e164 MY-TABLE
  map 10 to 20
  map 11 to 21
  map 12 to 22
  map 13 to 23
```

To remove the first two entries from the table enter the following commands:

```
node(cfg)#context cs
node(ctx-cs)[switch]#mapping-table MY-TABLE
node(map-tab)[switch.MY-TABLE]#no map 10
node(map-tab)[switch.MY-TABLE]#no map 11
```

The resulting running-config is:

```
mapping-table called-e164 to called-e164 MY-TABLE
  map 12 to 22
  map 13 to 23
```

Procedure: To delete an entire mapping table

Mode: Context CS

Step	Command	Purpose
1	<code>node(ctx-cs)[switch]#no mapping-table <i>table-name</i></code>	Delete the mapping table <i>table-name</i> . Note: You do not have to enter the type of the table when just deleting it. The type must only be specified when creating a table.

Example: Remove an entire mapping table

```
node(cfg)#context cs
node(ctx-cs)[switch]#no mapping-table MY-TABLE
```

Creating Complex Functions

Complex functions allow combining mapping tables, which need to be executed in sequence. This is useful if, for example, the calling and the called party number have to be modified in the same step. Complex function names can be any arbitrary string.

Procedure: To create a complex number manipulation function

Mode: Context CS

Step	Command	Purpose
1	<code>node(ctx-cs)[switch]#complex-function <i>function-name</i></code>	Create a complex function <i>function-name</i> .
2	<code>node(func)[function-name]#execute <i>function</i></code> or <code>node(func)[function-name]#execute <i>index function</i></code>	Add or inserts an entry to the complex function. <i>function</i> can be another complex function or a mapping table that shall be executed. Note: Unlike routing and mapping tables, complex functions are ordered lists of entries, which means that the entries are executed in the order of appearance. You can optionally specify an <i>index</i> of where to insert the function.
3		Repeat step 2 to add lines for additional functions to execute.

Example: Create a complex function

```
node(cfg)#context cs
node(ctx-cs)[switch]#complex function CARRIER-TO-LOCAL
node(func)[switch.CARRIER~]#execute 1 TRUNCATE-3-CNPN
node(func)[switch.CARRIER~]#execute 2 DDI-TO-PISN
```

Deleting Complex Functions

To remove individual complex functions you can use the **no** form of the **complex function** command. Alternatively you can remove specific entries of a complex function by entering the complex function configuration mode and use the **no** form of the **execute** command.

Procedure: To delete an entry from a complex function

Mode: Context CS

Step	Command	Purpose
1	<code>node(ctx-cs)[switch]#complex-function <i>function-name</i></code>	Enter the complex function from which you want to remove an entry.
2	<code>node(func)[switch.<i>table-name</i>]#no execute <i>index</i></code>	Remove the entry with the specified index.
3		Repeat Step 2 to remove additional entries.

Example: Remove entries from a complex function

The running-config shows the following complex function:

```
complex function MY-FUNC
  execute 1 MAP1
  execute 2 MAP2
  execute 3 MAP3
  execute 4 MAP4
```

To remove the first two entries from the complex function enter the following commands. Pay attention on the index. When removing the first entry, the MAP2 function becomes entry with index 1. Thus you have to specify index 1 twice to remove the first two entries:

```
node(cfg)#context cs
node(ctx-cs)[switch]#complex-function MY-FUNC
node(func)[switch.MY-FUNC]#no execute 1
node(func)[switch.MY-FUNC]#no execute 1
```

The resulting running-config is:

```
complex function MY-FUNC
  execute 1 MAP3
  execute 2 MAP4
```

Procedure: To delete an entire complex function

Mode: Context CS

Step	Command	Purpose
1	<code>node(ctx-cs)[switch]#no complex-function function-name</code>	Delete the complex function <i>function-name</i> .

Example: Remove an entire complex function

```
node(cfg)#context cs
node(ctx-cs)[switch]#no complex-function MY-FUNC
```

Digit Collection & Sending-complete Behavior

Sending-complete

The call-router can be configured how to handle address-complete indications (e.g. ISDN Sending-Complete IE). On ISDN and SIP interfaces, an **address-complete-indication** command controls the tunneling of **address-complete** indications separately for incoming and outgoing calls.

Each interface can be configured to pass transparently address-complete indications, or to explicitly insert or remove it for incoming and outgoing calls. In addition the behavior of the call-router can be configured.

Some call signaling protocols allow a user to dial a destination by using the overlap-sending procedure. These protocols include analog telephony (FXS/FXO) and ISDN. ISDN support address-complete indication using the Sending-Complete Information Element. Other protocols wait on a timeout or send a terminating character, e.g. pound (#), to indicate address-completion. The following commands control the tunneling of address-complete indications from the signaling protocol to the internal call-control representation and vice-versa. In addition, the context cs introduce command extensions that control the address-completion handling in the internal call-router.

Ingress interface

On the ingress interface (ISDN or SIP) the **address-complete-indication accept <type>** command configures how to map the address-complete indication of the signaling-protocol to the internal call-control. There the indication can be used for call-routing (see below) or mapped again to an address-complete indication on an egress interface (see below).

The possible values for the type argument are:

- **transparent:** Transparently passes an address-complete indication (e.g. ISDN Sending Complete Information Element) to the call-control.
- **set:** Always sets the address-complete indication flag towards the call-control even when no such indication is received from the calling party. This configuration can be used to disable overlap-sending on an interface.
- **clear:** Never signals the address-complete indication towards the call-control even when the indication is received from the calling party. This configuration may be used in rare cases to solve interoperability problems with attached PBXs.

Some interface types do not support all of the mentioned arguments. The following table shows the supported address-complete indication conversion types and the default setting for each interface type:

Interface Type	Transparent	Set	Clear
ISDN	supported; default	supported	Supported
SIP	not supported	supported	supported; default

Call-router

The call-router is extended to be able to set the address-complete indication or append a terminating-character in some circumstances. This includes:

- Setting the address-complete indication when the digit-collection timeout elapses
- Appending the terminating-character when the digit-collection timeout elapses
- Filtering-out the terminating-character and optionally set the address-complete indication
- Setting the address-complete indication when a called-party number matches a fully specified call-router rule (\$-terminated entry).
- Appending the terminating-character when the called-party number matches a fully specified call-router rule (\$-terminated entry).

The command **digit-collection timeout <secs>** has been extended with two optional arguments that specify whether a terminating-character is appended or the address-complete indication set when the digit-timeout elapses. The command syntax is

[no] digit-collection timeout <secs> [append-terminating-char] [set-address-complete-indication]

The extensions configure the action(s) to be performed when the digit-collection timeout elapses. The digit-collection timeout runs when a call is routed to a called-e164 routing-table of which a T-terminated rule is selected. Two actions are available; both can be enabled independently. The **append-terminating-char** action appends the configured terminating-character when the timeout elapses. The **set-address-complete-indication** action sets the address-complete indication. Whether or not the egress interface propagates the address-complete indication depends on the interface configuration (see below).

The following table shows the different digit-collection timeout configurations and their effect on a T-terminated route when the digit-collection timeout elapses. Important settings are marked bold.

Digit Collection Timeout Configuration			Ingress Properties		Egress Properties (Result of call-routing)	
Timeout	append-terminating-char	set-address-complete-indication	called-e164	Address-Complete Indication	called-e164	Address-Complete Indication

Digit Collection Timeout Configuration			Ingress Properties		Egress Properties (Result of call-routing)	
no	no	no	123	false	no call	
		yes		true		
		no		false		
		yes		true		
	yes	no		false		
		yes		true		
		no		false		
		yes		true		
yes	no	no	false	123	false	
		yes	true		true	
		no	false		true	
		yes	true		true	
	yes	no	false	123#	false	false
		yes	true		true	true
		no	false		true	
		yes	true		true	

The command **digit-collection terminating-char <char>** has been extended with two optional arguments that specify whether the terminating-character is re-appended or the address-complete indication is set when a digit-collection timeout is stopped by the user sending the terminating-character. The new command syntax is:

[no] digit-collection terminating-char <char> [append-terminating-char] [set-address-complete-indication]

The extensions configure what action(s) shall be performed when the digit-collection timeout is stopped by the reception of a terminating-character. Normally, without specifying an action, the received terminating-character is removed from the called-party number. The **append-terminating-char** action re-appends the terminating-character. The **set-address-complete-indication** action sets the address-complete indication. Whether or

not the exit interface propagates the address-complete indication depends on the interface configuration (see below).

The next table shows the different digit-collection terminating-character configurations and their effect on a T-terminated route when the terminating-character is received.

Digit Collection Terminating-Char Configuration			Ingress Properties		Egress Properties (Result of call-routing)			
Terminating-Char	append-terminating-char	set-address-complete-indication	called-e164	Address-Complete Indication	called-e164	Address-Complete Indication		
No	no	no	123#	False	123#	false		
		yes		true		true		
		no		false		false		
		yes		true		true		
	yes	no		false		false		
		yes		true		true		
		no		false		false		
		yes		true		true		
Yes	no	no	123	false	123	false		
		yes		true		true		
		no		false		true		
		yes		true		true		
	yes	no		no	123#	false	123#	false
				yes		true		true
				no		false		true
				yes		true		true
		yes		no		false	false	
				yes		true	true	
				no		false	true	
				yes		true	true	

The command **digit-collection full-match** has been introduced. The syntax is:

digit-collection full-match [append-terminating-char] [set-address-complete-indication]

This command configures what action(s) shall be performed when a dollar-(\$)-terminated rule is selected in a called-e164 routing-table. We call such an entry a fully specified number entry. The **append-terminating-char** action appends the terminating-character; while the **set-address-complete-indication** action sets the address-complete indication. Whether or not the egress interface propagates the address-complete indication depends on the interface configuration (see below).

Egress interface

On the egress interface (ISDN) the **address-complete-indication emit <type>** command configures how the internal representation of the address-complete indication shall be mapped to the representation of the signaling-protocol.

The possible values for the type argument are:

- **transparent:** Transparently passes an address-complete indication to the signaling-protocol (e.g. ISDN by sending a Sending Complete Information Element).
- **set:** Always sends a Sending Complete Information Element with the SETUP message. This configuration can be used to disable overlap-sending on an interface.
- **clear:** Never sends a Sending Complete Information Element. This configuration may be used in rare cases to solve interoperability problems with attached PBXs, e.g. when the attached PBX does not support the Sending Complete Information Element.

Some interface types do not support all arguments. SIP does not support this configuration at all, because SIP does not support overlap dialing. The following table shows the supported address-complete indication conversion types and the default setting for each interface type:

Interface Type	Transparent	Set	Clear
ISDN	supported; default	supported	Supported
SIP	—	—	—

The following procedure demonstrates how to disable overlap-sending for incoming SIP calls. SIP does not provide an overlap-dialing procedure; so for most applications, address-complete indications should be cleared.

Mode: context cs / interface sip

Step	Command	Purpose
1	<i>node(if-sip)[if-name]#address-complete-indication clear</i>	Clear the address-complete indication for all incoming calls over this interface

The following procedure demonstrates how to create a routing-table that allows overlap dialing. When the timeout elapses or when the terminating-character is received, the address-complete indication shall be set toward the egress interface.

Mode: context cs

Step	Command	Purpose
1	<i>node(cts-cs)[ctx-name]#routing-table called-e164 <tab-name></i>	Creates a routing-table that examines the called-party number
2	<i>node(rt-tab)[tab-name]#route T dest-interface <if-name></i>	Routes any number (after the waiting for digits) to the egress interface

Step	Command	Purpose
3	<code>node(rt-tab)[tab-name]#exit</code>	Leaves the routing table configuration mode and returns to the context cs configuration mode
4	<code>node(cts-cs)[ctx-name]#digit-collection timeout 5 set-address-complete-indication</code>	Configures the digit-collection timeout to 5 seconds and sets the address-complete indication if the timeout elapses
5	<code>node(cts-cs)[ctx-name]#digit-collection terminating-char # set-address-complete-indication</code>	Configures the terminating-character that can stop the digit-collection timeout and immediately place the call to '#' and sets the address-complete indication if the character is received

Creating Call Services

Routing tables, mapping tables and complex functions only manipulate address properties of a call (like the called party number). A call service is another call router entity that actively accesses the state of a call. A call service is able to spawn other calls or merge existing calls together. This allows building services like hunt groups etc.

You can view a call service as a virtual endpoint that accepts a call and creates new calls to other destinations. A call that is routed to a call service is not served by a human being but by a machine that accepts the call and performs needed actions.

The hunt group service, for example, accepts a call that is routed to it and spawns a second call that is placed to the first final destination. If this destination is not reachable, another destination is tried until one of the configured destinations accept the call.

Creating a Hunt Group Service

A hunt group service hunts an incoming call to multiple interfaces. [figure 93](#) shows an example scenario where a call from a SIP interface is first processed by several tables. The second table decides that the call must be forwarded to the PSTN. The device is connected to the PSTN over four BRIs, which are bound to the context CS

interfaces IF-BRI0 up to IF-BRI3. All four ISDN interfaces lead to the same provider. Since the call router does not know the load on the BRIs, it has to be able to try BRI0 and, if BRI0 already serves two calls, use BRI1, and so on.

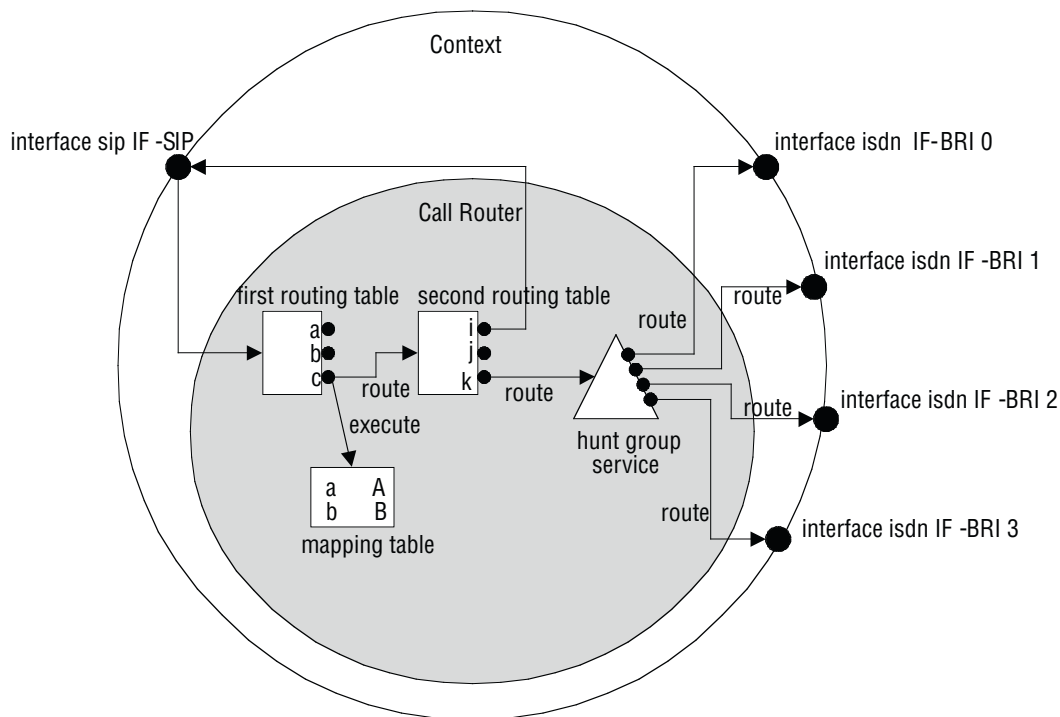


Figure 93. Hunt group service

The hunt group service accepts a call routed to it by a routing table or directly from an interface and creates another call that is offered to one of the configured destination interfaces.

The interface tried first (IF-BRI0) may drop the call telling the service that the interface has no resources to handle the call (e.g. no circuit channel available). Note that only the call between the hunt group and the destination interface is dropped, while the original call between the SIP interface and the hunt group service remains connected.

The hunt group then decides to try the next destination (IF-BRI1), which in turn also drops the call due to unavailable resources. In our example the hunt group then tries the third and eventually the fourth destination. When an interface accepts a call, the interface hunting is complete and the hunt group service merges the original with the new call to the interface that accepted the call.

You can influence the algorithm of the hunt group by several configuration commands. You can specify whether the hunt group shall always start with the same destination interface or whether it shall immediately try the next one in a round-robin fashion. This is called cyclic operation mode.

You can specify a timeout after which the next destination interface is tried when there is no answer at all from the destination interface.

You can specify drop causes that trigger hunting for the next destination. All other causes (e.g. user busy) will drop the original call.

Note You can hunt a call over different interface types, not only over ISDN interfaces. You can, e.g. create a hunt group to try to call over a SIP interface and, if this call fails, do a fallback to an ISDN interface.

Procedure: To create and configure a hunt group service

Mode: Context CS

Step	Command	Purpose
1	<code>node(ctx-cs)[switch]#service hunt-group service-name</code>	Creates a new hunt group service and enters hunt group configuration mode.
2	<code>node(svc-hunt)[service-name]#cyclic</code> or <code>node(svc-hunt)[service-name]#no cyclic</code>	Configure the hunt group for cyclic operation mode. Subsequent calls try another first destination in a round-robin method. Default is not to use cyclic mode – always to start with the first configured destination. Note: When you use the hunt-group for a fallback scenario, you must switch off cyclic operation mode.
3	<code>node(svc-hunt)[service-name]#time-out timeout</code>	Configures a timeout in seconds after which the next destination is tried when the current destination does not answer at all. (Some interface (e.g. SIP) may wait an arbitrary long time until an answer is returned.) Default is not to use a timeout.
4	<code>node(svc-hunt)[service-name]#drop-cause cause</code> or <code>node(svc-hunt)[service-name]#no drop-cause cause</code>	Enables or disables another drop cause to the list. When an interface has a problem placing a call to the final destination it drops the call specifying a drop cause (e.g. user busy, no resource available). Some drop causes drop the original call while other causes trigger the hunt for another destination. This command can be used to configure the hunt behavior on destination call drop. See the list below for a summary of all available drop causes and their default state.
5	<code>node(svc-hunt)[service-name]#route call dest-interface interface-name</code> or <code>node(svc-hunt)[service-name]#route call dest-table table-name</code> or <code>node(svc-hunt)[service-name]#route call dest-service service-name</code>	Adds a route to a destination. This is the destination that is tried during hunt group's interface hunting. The destination can either be an interface or you can route the call again to a routing table or directly to another service. This allows you cascading services.
6		Repeat step 5 to add additional hunting destinations.

When a destination interface drops the call, the hunt group service has to decide whether this is because the interface is not able to handle the call (e.g. no bearer channel available) or whether the destination user does not want or is not able to communicate (e.g. user busy). In the first case, the hunt group service hunts for the next destination interface, while in the second case the original call is dropped with the same cause.

The following table lists all drop causes and specifies whether the cause is used for hunting the next destination or dropping the original call. The behavior can be configured for each hunt group individually for each cause using the **drop-cause** command in the hunt group service mode.

Table 33. Hunt group drop causes

Class	Cause	Default Behavior of the Hunt Group Service	Description
Normal Event	unallocated-number	Drop original call	The number is sent in the correct format. However, the number is not assigned to the destination equipment.
	no-route-to-network	Drop original call	The destination is asked to route the call through an unrecognized network. This cause indicates that the equipment sending this cause has received a request to route the call through a particular transit network, which it does not recognize. The equipment sending this cause does not recognize the transit network either because the transit network does not exist or because that particular network, while it does exist, does not serve the equipment that is sending this cause.
	no-route-to-destination	Drop original call	The call routes through an intermediate network that does not serve the destination address. The called user cannot be reached because the network through which the call has been routed does not serve the destination desired.
	channel-unacceptable	Drop original call	The service quality of the specified channel is insufficient to accept the connection. The call attempt failed because the channel cannot be used.
	call-awarded	Drop original call	The user assigns an incoming call that is connecting to an already established call channel.
	normal-call-clearing	Drop original call	Normal call clearing occurs. This cause indicates that the call is being cleared because one of the users involved in the call has requested that the call be cleared. Under normal situations, the source of this cause is not the network.
	user-busy	Drop original call	The called system acknowledges the connection request but cannot accept the call because all channels are in use. It is noted that the user equipment is compatible with the call.

Table 33. Hunt group drop causes (Continued)

Class	Cause	Default Behavior of the Hunt Group Service	Description
Normal Event (Cont.)	no-user-responding	Drop original call	The connection fails because the destination does not respond to the call. This cause is used when a user does not respond to a call establishment message with either an alerting or a connect indication with the prescribed period of time allocated.
	no-answer-from-user	Drop original call	The destination responds to the connection request but fails to complete the connection within the prescribed time. This cause is used when the user has provided an alerting indication but has not provided a connect indication within a prescribed period of time.
	subscriber-absent	Drop original call	The remote device you attempted to reach is unavailable and has disconnected from the network.
	call-rejected	Drop original call	The destination can accept the call but rejects it for an unknown reason. This cause indicates that the equipment sending this cause does not wish to accept this call, although it could have accepted the call because the equipment sending this cause is neither busy nor incompatible.
	number-changed	Drop original call	The number used to set up the call is not assigned to a system. This cause is returned to a calling user when the called party number indicated by the calling user is no longer assigned.
	non-selected-user-clearing	Drop original call	The destination can accept the call but rejects it because it is not assigned to the user.
	destination-out-of-order	Drop original call	The destination cannot be reached because of an interface malfunction, and a signaling message cannot be delivered. This can be a temporary condition, but it could last for an extended period.
	invalid-number-format	Drop original call	The connection fails because the destination address is presented in an unrecognizable format, or the destination address is incomplete.
	facility-rejected	Drop original call	The network cannot provide the facility requested by the user.

Table 33. Hunt group drop causes (Continued)

Class	Cause	Default Behavior of the Hunt Group Service	Description
Normal Event (Cont.)	response-to-status-inquiry	Drop original call	The status message is generated in direct response to receiving a status inquiry message.
	normal-unspecified	Drop original call	Reports the occurrence of a normal event when no standard cause applies.
Resource Unavailable	no-circuit-channel-available	Hunt for next destination	The connection fails because no appropriate channel is available to take the call.
	network-out-of-order	Hunt for next destination	The destination cannot be reached because of a network malfunction, and the condition can last for an extended period. An immediate reconnect attempt will probably fail.
	temporary-failure	Hunt for next destination	An error occurs because of a network malfunction. The problem will be resolved shortly.
	switching-equipment-congestion	hunt for next destination	The destination cannot be reached because the network switching equipment is temporary overloaded.
	access-info-discarded	Hunt for next destination	The network cannot provide the requested access information. This cause indicates that the network could not deliver access information to the remote user as requested.
	circuit-channel-not-available	Hunt for next destination	The equipment cannot provide the requested channel for an unknown reason.
	resources-unavailable	Hunt for next destination	The requested channel or service is unavailable for an unknown reason.
Service or Option Not Available	qos-unavailable	Drop original call	The network cannot provide the requested quality of service.
	facility-not-subscribed	Drop original call	The remote equipment supports the requested supplementary service by subscription only. This cause indicates that the network could not provide the requested supplementary service because the user has not completed the necessary administrative arrangements with its supporting networks.
	bearer-capability-not-authorized	Drop original call	The user requests a bearer capability the network provides, but the user is not authorized to use it. This can be a subscription problem.

Table 33. Hunt group drop causes (Continued)

Class	Cause	Default Behavior of the Hunt Group Service	Description
Service or Option Not Available (Cont.)	bearer-capability-not-available	Drop original call	The network normally provides the requested bearer capability, but it is unavailable at the present time. This can be due to a temporary network problem or subscription problem.
	service-or-option-not-available	Drop original call	The network or remote equipment cannot provide the requested service option for an unspecified reason. This can be a subscription problem.
Service or Option Not Implemented	bearer-capability-not-implemented	Drop original call	The network cannot provide the bearer capability requested by the user.
	channel-type-not-implemented	Drop original call	The network or the destination equipment does not support the requested channel type.
	facility-not-implemented	Drop original call	The remote equipment does not support the requested supplementary service.
	only-restricted-digital-available	Drop original call	The network cannot provide unrestricted digital information bearer capability. This cause indicates that a device has requested an unrestricted bearer service but the equipment sending this cause only supports the restricted version of the requested bearer capability.
	service-or-option-not-implemented	Drop original call	The network or remote equipment cannot provide the requested service option for an unspecified reason. This can be a subscription problem.
Invalid Message	invalid-call-reference	Drop original call	The remote equipment receives a call with a call reference value that is not currently in use.
	channel-does-not-exist	Drop original call	The receiving equipment is requested to use a channel that is not activated on the interface for calls. This cause indicates that the equipment sending this cause has received a request to use a channel not activated on the interface for a call.
	call-identity-does-not-exist	Drop original call	This cause indicates that a call resume has been attempted with a call identity, which differs from that in use for any presently suspended call.

Table 33. Hunt group drop causes (Continued)

Class	Cause	Default Behavior of the Hunt Group Service	Description
Invalid Message (Cont.)	call-identity-in-use	Drop original call	This cause indicates that the network has received a call suspends request. The call suspend request contained a call identity which is already in use for a suspended call within the domain of interfaces over which the call might be resumed.
	no-call-suspended	Drop original call	The network receives a call resume request when there is not a suspended call pending. This can be a transient error that will be resolved by successive call retries.
	call-has-been-cleared	Drop original call	The network receives a call resume request. This call resume request contains a call identity that once indicated a suspended call. However, the suspended call was cleared either by time-out or by the remote user.
	incompatible-destination	Drop original call	Indicates that an attempt is made to connect incompatible equipment.
	invalid-transit-network	Drop original call	This cause indicates that a transit network identification of an incorrect format was received.
	invalid-message	Drop original call	Received an invalid message with no standard cause.

Table 33. Hunt group drop causes (Continued)

Class	Cause	Default Behavior of the Hunt Group Service	Description
Protocol Error	mandatory-ie-missing	Drop original call	The receiving equipment receives a message that does not include one of the mandatory information elements. This cause indicates that the equipment sending this cause has received a message that is missing a call property that must be present in the message before that message can be processed.
	message-type-not-implemented	Drop original call	The receiving equipment receives an unrecognized message, because the message type is invalid or the message type is valid but not supported.
	message-type-not-state-compatible	Drop original call	The remote equipment receives an invalid message with no standard cause. This cause indicates that the equipment sending this cause has received a message such that the procedures do not indicate that this is a permissible message to receive while in the current call state.
Protocol Error (Cont.)	ie-does-not-exist	Drop original call	The remote equipment receives a message that includes information elements or call properties that are not recognized.
	invalid-ie-contents	Drop original call	The remote equipment receives a message that includes invalid information in the information element or call property.
	recovery-on-timer-expiry	Drop original call	Your call was not completed, probably because an error occurred.
	protocol-error	Drop original call	An unspecified protocol error with no other standard cause occurred.
Interworking	interworking	Drop original call	An event occurs, but the network does not provide causes for the action it takes. The precise problem is unknown.

Example: Create a hunt group service

This example shows how to configure the hunt group service as shown in figure 93 on page 606.

```
node(cfg)#context cs
node(ctx-cs)[switch]#service hunt-group HUNT-BRI
node(svc-hunt)[HUNT-BRI]#cyclic
node(svc-hunt)[HUNT-BRI]#timeout 6
node(svc-hunt)[HUNT-BRI]#route dest-interface IF-BRI0
node(func)[HUNT-BRI]#route dest-interface IF-BRI1
```

```
node(func)[HUNT-BRI]#route dest-interface IF-BRI2
node(func)[HUNT-BRI]#route dest-interface IF-BRI3
```

Defining the Hunt-group Behavior for Inband Information

This command defines the behavior of a Hunt-Group service if a hunt-peer indicates the availability of inband information. Choose from three options:

- **Transparent:** This is the default behavior. Inband information is always passed through from hunt-peers to the call originator.
- **Block:** Inband information is never passed through from hunt-peers to the call originator.
- **Succeed:** If a hunt-peer announces inband information either in a provisional state or in disconnecting state, hunting is going to be stopped and the call is switched through the current destination. This happens even the peer disconnects the call with a cause that is configured as an indication to proceed hunting.

Mode: Service Hunt-Group

Step	Command	Purpose
1	<code>[node](svc-hunt)[name]# inband-information { transparent block succeed }</code>	Defines the Hunt-Group behavior if a destination announces inband information. Default: transparent

SIP handover on RTP loss with Hunt-Group

So far, a hunt-group service could only be used to provide a fallback to different interfaces at the call-establishment phase. For example, to configure alternate paths for SIP calls, the administrator has to configure multiple SIP interfaces and route calls to them from a hunt-group service. The hunt-group service then hunted for a successful connection over the configured SIP interfaces until the call was established.

The new 'call-handover' feature, configured on the hunt-group service, allows for seamless handover during a SIP call. That is, if the RTP connection is detected broken, the original call is replaced with another call over another SIP interface. For this to work, the hunt-group remains in the original call and detects RTP connection-broken events. In case of such an event, the hunt-group establishes a new replacement call to the next configured destination (i.e. SIP interface) and, if connected successfully, switches the RTP stream over to the new call.

Note This feature currently only works over SIP and only if the SIP peer UA supports replacing the media stream of the original call with that of the replacement call using one of the methods described below:

We support two methods to tell the SIP peer UA that the INVITE message replaces another call:

- With a Replaces header: The replacement call contains a 'Replaces' header according to RFC 3891, which contains the Call-ID, as well as the From and To header tags of the original call:

```
Replaces: cab569ce83a9026ab;from-tag=1854a30e67;to-tag=6b357f4fa3
```

- With Patton-specific headers: The replacement call contains two proprietary X-headers, one signalling the handover condition and another one to provide the value of the Call-ID header of the original call.

```
X-Patton-Call-Failure: true
X-Patton-Call-ID: cab569ce83a9026ab
```

- No signaling: No specific information about the original call is added to the replacement call.

Note The users participating in the call do not hear each other between the RTP connection being broken and the media of the replacement call being fully connected; this usually takes a couple of seconds. Also note that the original call with the broken media connection will be terminated a few seconds after the replacement call is established.:



This feature requires the 'sip-handover' license.

Mode: context cs / service hunt-group

Step	Command	Purpose
1	<code>node(svc-hunt)[<name>]# call-handover on-media-loss</code>	Configures the hunt-group to stay in the call and issue a replacement call when a loss of the RTP stream is detected. The conditions for detecting an RTP connection-loss is configured in the voip profile (media loss-detection command). Default: disabled. Requires license: sip-handover.

Mode: context cs / interface sip

Step	Command	Purpose
1	<code>node(if-sip)[<name>]# [no] call-handover signaling (replaces-header patton-header)</code>	Configures the method of tagging the replacement call with information about the original call. Default: replaces-header.

Mode: profile voip

Step	Command	Purpose
1	<code>node(pf-voip)[<name>]# media loss-detection <ms></code>	Configures after which time in milliseconds without receiving an RTP packet the media connection is considered broken. Default: 1000 ms. Range: 1000-10000 ms.

Creating a Distribution Group Service

A distribution group service distributes a call to multiple destination interfaces. [figure 94](#) shows an example scenario where a call from a SIP interface is first processed by several tables. The second table decides that the call must be forwarded to phones that are connected to various ISDN interfaces. The distribution now lets all four phones ring at the same time.

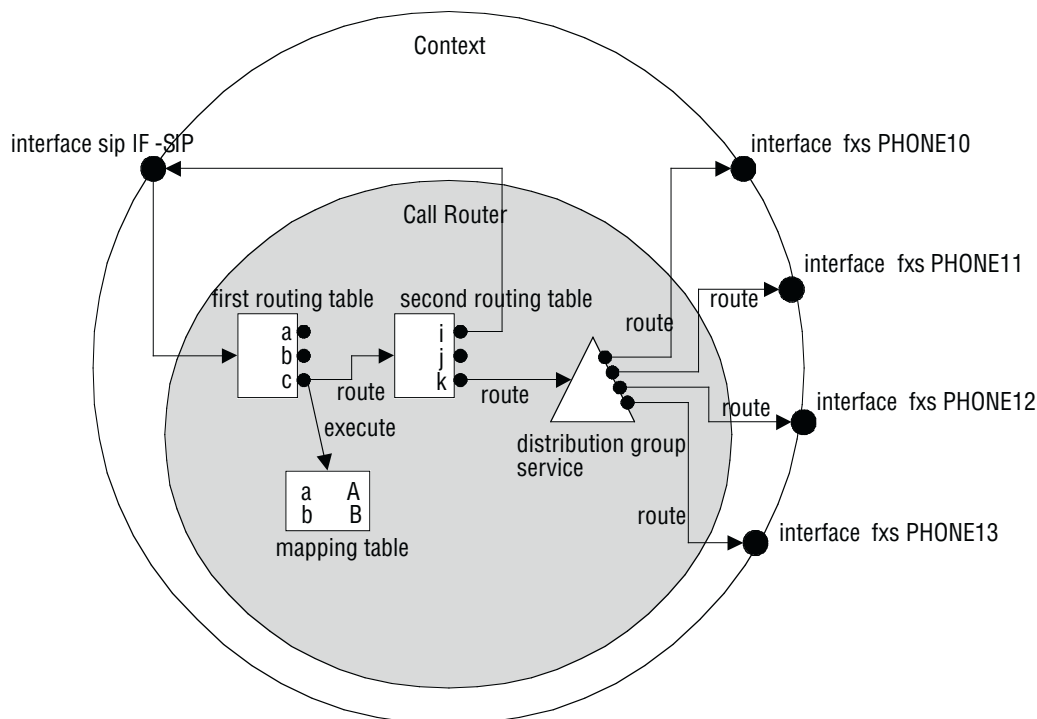


Figure 94. Distribution group service

The distribution group service accepts a call routed to it by a routing table or directly from an interface and creates four other calls that are offered to each of the configured destination interfaces.

All phones connected to the ISDN interfaces (*PHONE10–PHONE13*) start ringing. Eventually one of the phones (e.g. *PHONE10*) goes off-hook. The other three calls to interfaces *PHONE11*, *PHONE12*, and *PHONE13* are immediately dropped and the phones on these interfaces stop ringing. Now the distribution service is no longer needed. Thus the service merges the original call to the accepted destination call to interface *PHONE10*.

You can configure how the distribution algorithm works in many ways. You can specify the maximum number of destination interfaces that are called at the same time. Then you can specify a timeout after which a next destination is added to the destination calls. This makes it possible to configure a scenario where a call is offered to two destinations, and (when no one answers) stop ringing the first phone but try another third destination.

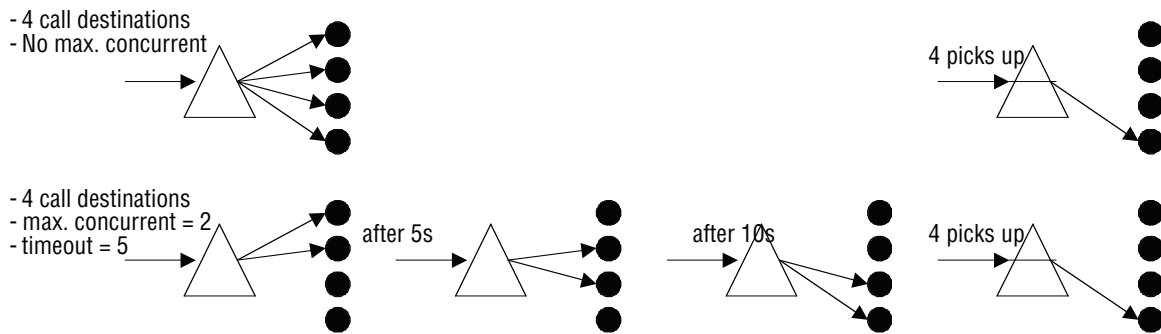


Figure 95. Distribution group service examples

Procedure: To create and configure a distribution group service

Mode: Context CS

Step	Command	Purpose
1	<code>node(ctx-cs)[switch]#service distribution-group service-name</code>	Creates a new distribution group service and enters distribution group configuration mode.
2	<code>node(svc-hunt)[service-name]#cyclic</code>	Configure the distribution group for cyclic operation mode. Subsequent calls try another first destination in a round-robin method. Default is not to use cyclic mode – always to start with the first configured destination(s).
3	<code>node(svc-hunt)[service-name]#max – concurrent max-concurrent</code>	Configures how many destinations shall be called at the same time. If you also configure a timeout, the first call is cleared and an additional call is made after that timeout. Thus only the specified number of destinations is ringing at the same time.
4	<code>node(svc-hunt)[service-name]#time-out timeout</code>	Configures a timeout in seconds after which one destination is dropped and a next destination is called.
5	<code>node(svc-hunt)[service-name]#route call dest-interface interface-name</code> OR <code>node(svc-hunt)[service-name]#route call dest-table table-name</code> OR <code>node(svc-hunt)[service-name]#route call dest-service service-name</code>	Adds a route to a destination. This is the interface, table or service that is tried to call during the distribution group's attempt to make calls to the destinations. The destination can either be an interface or you can route the call again to a routing table or directly to another service. This allows you cascading services.
6		Repeat step 5 to add additional hunting destinations.

Note If you specified the maximum number of concurrent destinations and the distribution group tried each destination, the final destinations ring until someone picks up one of the phones.

Note It does not make sense to configure the maximum number of concurrent destinations but no timeout, though the software does not prevent this configuration.

Distribution-Group Min-Concurrent Setting

A new command in the call-control's distribution-group service lets the user specify how many of the configured call destinations should be tried first: **min-concurrent <number>**. Together with the **max-concurrent** and **timeout** commands, you can configure the behavior of the call distribution. The distribution-group starts with *min-concurrent* calls and after the *timeout* one destination call is added until *max-concurrent* is reached. Then, again after the timeout, a call to the next destination is placed while the first destination call is dropped.

Mode: service distribution-group <name>

Step	Command	Purpose
1	<code>[name](svc-dist)[name]# min-concurrent <number></code>	Sets the minimum number of concurrent calls.

Example: To configure a distribution group that first rings on the phone#1 and then, after 5 seconds, on phone#1 and phone#2, enter the following commands:

```
node(svc-dist)[service-name]#min-concurrent 1
node(svc-dist)[service-name]#max-concurrent 2
node(svc-dist)[service-name]#timeout 3
node(svc-dist)[service-name]#route call dest-interface PHONE1
node(svc-dist)[service-name]#route call dest-interface PHONE2
```

Call-router 'limiter' Service

The call-router 'limiter' service limit offers a flexible technique to limit the maximum number of concurrent calls within the system. It also limits the call-setup rate within a system. Calls exceeding the defined limits can either be simply dropped or they can be handled differently. A call is counted as an active call as soon as the call-setup message reaches the limiter service. The call remains active until the signaling has completed tearing down the call.

In the figure is a call-router configuration which uses the limiter service to limit the maximum number of concurrent calls between SIP and ISDN to 20. In this scenario, if the limit is reached, any additional call received from sip will be dropped. If however an additional call arrives from ISDN, it will be forwarded to the special ISDN interface called 'voicemail'.

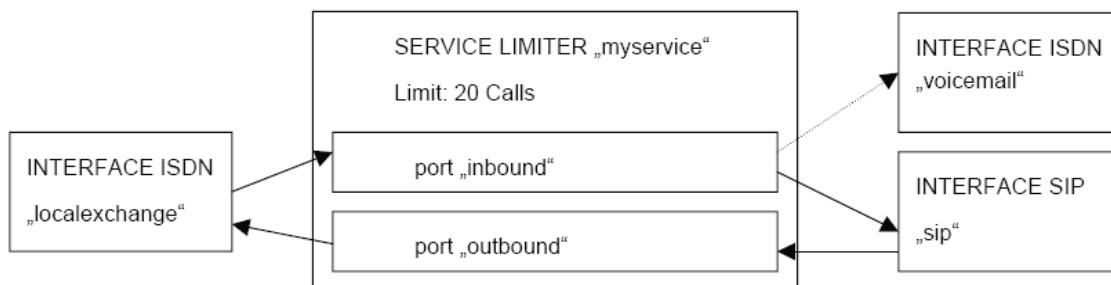


Figure 96. 'Limiter' service diagram

```

context cs switch
interface isdn localexchange
route call dest-service mylimiter.inbound
interface isdn voicemail
interface sip sip
bind gateway sip
route call dest-service mylimiter.outbound
service limiter mylimiter
max-calls 20
port inbound
route call dest-interface sip
exceed max-calls route call dest-interface voicemail
port outbound
route call dest-interface localexchange

```

Similarly the call-setup-rate could be limited by using the 'exceed max-call-rate' instead of the 'exceed max-calls' command in the limiter-port configuration mode. There is no limitation on the number of ports a limiter can have. You can create as many as you need for your application.

Priority Service

The service 'priority' can automatically free resources if a high priority call needs to be established while no resources are available. The service 'priority' can have multiple ports. You can assign a priority level for each port. This priority level defines the priority level of each call, which is received through the port. If a call with higher priority fails to be established, the service tries dropping lower priority calls to free resources for the higher priority call. Subsequently it tries to establish the higher priority call again. [Figure 97](#) on page 619 is a typical application for this service in which non-emergency calls are dropped to free resources for emergency calls.

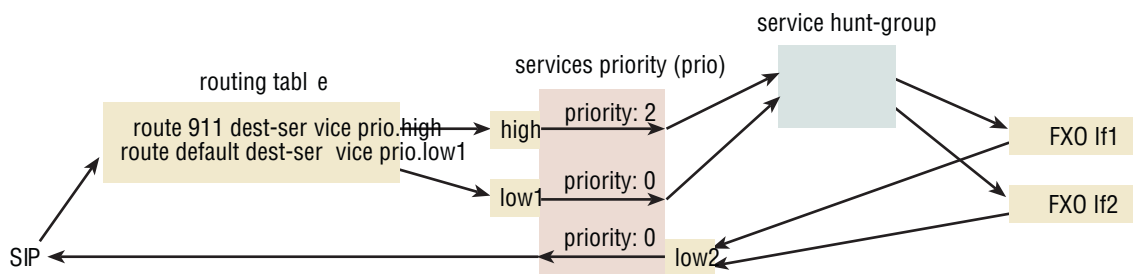


Figure 97. Priority service diagram

By default, the service drops any lower priority calls if a higher priority call fails. However, you have the option to limit the number of lower priority calls to be dropped if a higher priority call fails. It also blocks new lower priority calls for a configurable time after a higher priority call failed.

The following procedure demonstrates how to configure a priority service.

Mode: context cs

Step	Command	Purpose
1	node(ctx-cs)[ctx-name]# service priority <svc-name>	Create a priority service.
2	node(svc-prio)[svc-name]# port <port-name>	Create a port within the service.
3	node(port)[port-name]# route dest-....	Define the routing destination for calls received on this port.
4 (Optional)	node(port)[port-name]# priority	Define the priority for calls received through this port. (The default priority level is 0)
5	node(port)[port-name]# exit	Leave the port configuration mode
6		Repeat steps 2 to 5 for any additional calls you need to create.
7 (Optional)	node(svc-prio)[svc-name]# max-calls-to-drop <calls>	Define the maximum number of lower priority calls to be dropped if a higher priority call fails. (Default is to drop any lower priority calls.)
8 (Optional)	node(svc-prio)[svc-name]# quiesce-time <seconds>	Define the time for which lower priority calls are blocked after a higher priority call failed. (Default is 15 seconds.)
9 (Optional)	node(svc-prio)[svc-name]# retry-timeout <seconds>	Define the time for which the service waits after a higher priority call failed until it tries to establish it for a second time. This allows resources used by dropped lower priority calls to get available again. (Default is 3 seconds.)

Note Although this service improves the probability that higher priority calls can be established successfully, there is no guarantee that a higher priority task can be established successfully at any time.

CS Bridge Service—‘VoIP Leased Line’

The circuit switch (CS) bridge service provides the functional ability to create a leased line between two FXS ports, with the FXS ports on different Patton devices. The call is point-to-point in an *always connected* state, also known as *nailed up*. This Call Control service is called *Bridge services*.”

The application for this feature is when a constantly connected VoIP link is required between two FXS ports. It is as if you connected a regular telephone that is always off-hook to the FXS port. However it is not identical since the other end of the VoIP leased-line connection does not ring like in a PLAR application. There is no end-to-end call setup so no call-progress tones are required nor needed.

Here is another way to describe the application. A user has an FXO port in two remote locations and wants to connect these two locations over an IP network. Clearly the FXO port at a location must connect to an FXS port. But since the network between the two sites is IP, there needs to be a mapping of the information on the FXS–FXO link to a method of transporting the information over an IP network. Additionally, you do not want any ringing to occur when the connection is made; you simply want it to be connected, so the Patton devices

and IP network operate transparently. See [figure 98](#) to visualize the VoIP leased-line connection.

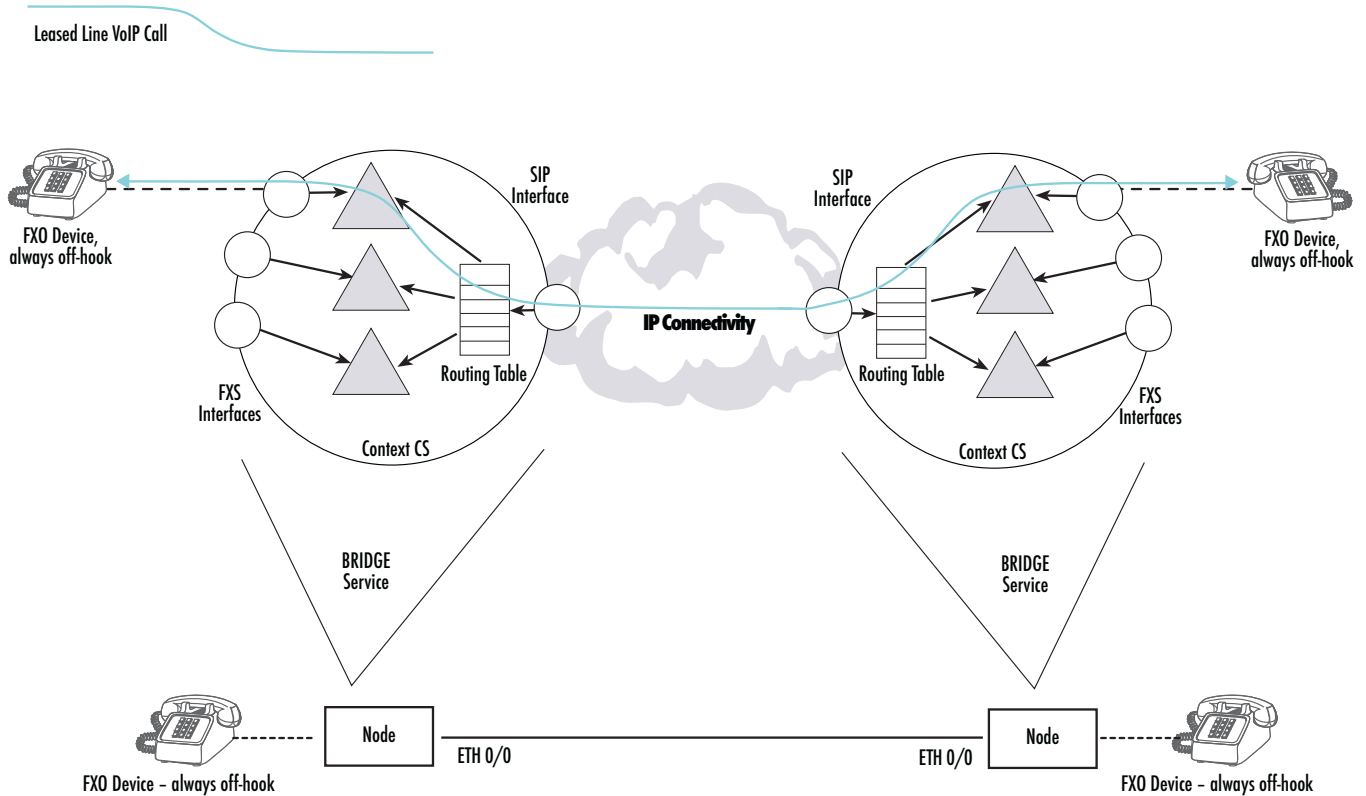


Figure 98. CS Bridge service—'VoIP Leased Line' diagram

Now we will describe the technical details and logical structure to implement this application. From the perspective of just one Patton device, the device makes two independent calls. These two calls are made from a logical structure, called a *Bridge Service*. The Bridge Service has two interfaces, the *listener* port and the *dialer* port (see [figure 99](#) on page 622). Each of these interfaces is responsible for one of the two independent calls. The lis-

tener port terminates the “FXS call” and the dialer port terminates the “RTP call.”

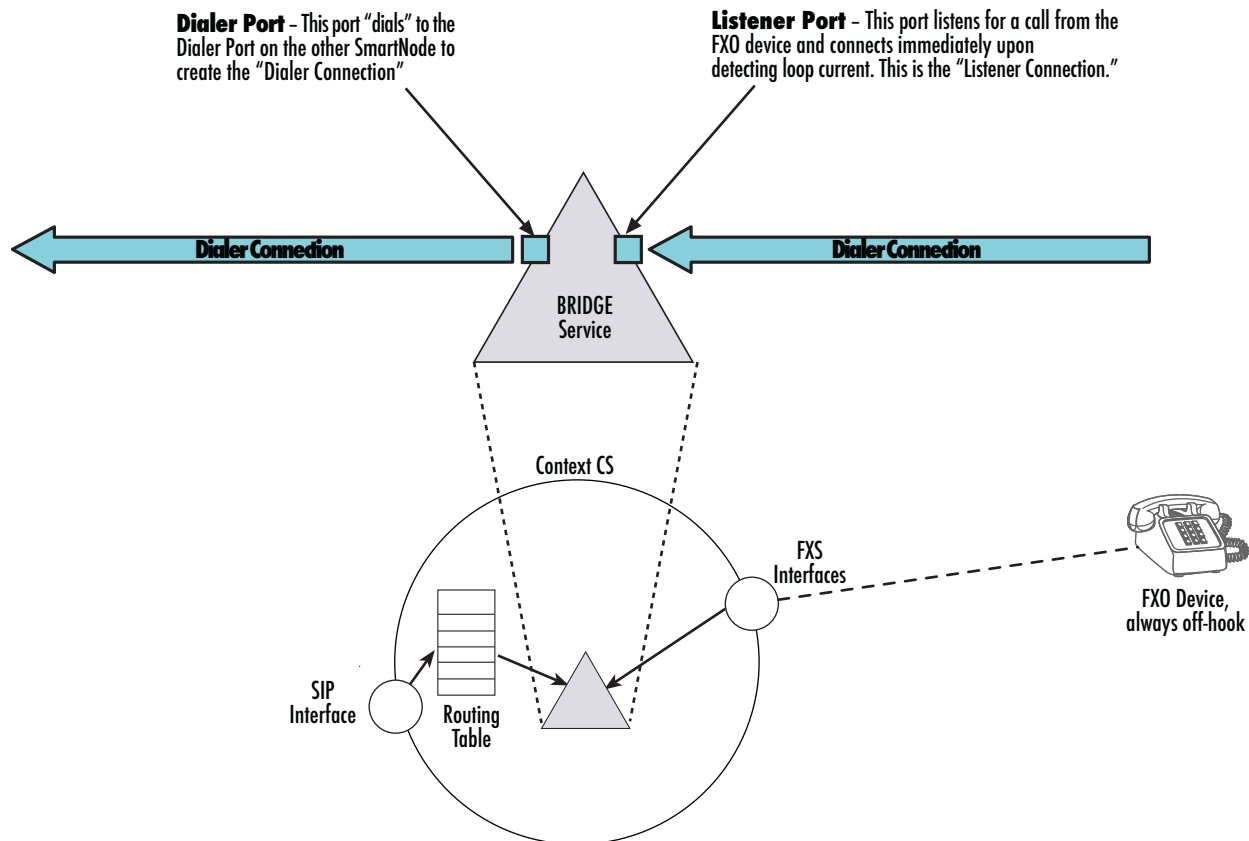


Figure 99. Bridge services diagram

The listener port interface listens to the FXS interface of the device for an active FXS–FXO connection. It recognizes an active connection by detecting current in the FXS–FXO loop, and the *FXS call* is established. The dialer port interface attempts to make and keep an RTP connection session or *call* with a dialer port in a remote device. It is called an *RTP call*. This connection is over the IP network. The listener port and the dialer port both try to keep their individual calls up and operating at all times. However if the listener port loses its connection (that is, its call), the dialer port does not disconnect its RTP call but remains connected to the other device’s dialer port. Similarly, if the local dialer port loses its connection with the remote device’s dialer port, the device’s listener port does not disconnect its FXS call but remains connected to the FXO device. Though the calls operate independently, they operate over a single data path from end-to-end.

The dialer port is bound to the Routing Table which is subsequently bound to either a SIP interface. The routing table makes the connection to the proper FXS interface.

Mode: Context CS

Step	Command	Purpose
1	<code>[name] (ctx-cs)[switch]#[no] service bridge BRIDGE1</code>	Enters the bridge service configuration mode / deletes a bridge service.
2	<code>[name] (svc-bridg)[BRIDGE1]#port DIALER</code>	Creates a port on the service that can accept or spawn calls (the max number of ports is currently limited to two)
3	<code>[name] (port)[DIALER]# dial persistent 123 dest-interface REMOTE</code>	Configures the port to actively dial the called-party "123" to the destination REMOTE. This connection is kept open until the service is shut down, or a "no dial" command is issued. If the remote side terminates the connection, the port tries to reconnect. As soon as the connection is established, the call is connected to all other calls present on the service.
4	<code>[name] (svc-bridg)[BRIDGE1]#port LISTENER</code>	Creates a port on the service that can accept or spawn calls (the max number of ports is currently limited to two)
5	<code>[name] (port)[DIALER]# no dial</code>	Without entering a "dial" command, or by specifying "no dial", this port does not dial, but listen for incoming calls. If a call comes in, it is automatically connected to all other calls present on the service.

Configuration Example:

```
context cs switch

routing-table called-e164 DISPATCH
  route 1 dest-service BRIDGE1.dialer
  ...

service bridge BRIDGE1
  port listener
    mode listening
    no shutdown
  port dialer
    mode dial-persistent 1 dest-interface REMOTE
    no shutdown

interface sip REMOTE
  bind gateway sip
  route call dest-table DISPATCH
  remote 172.16.32.37

interface fxs PHONE1
  route call dest-service BRIDGE1.listener
```

Configuring the Service Second-dialtone

If a call is routed into the service second-dialtone, the service establishes the call and plays a dial tone to the caller. When a call is entering the service it is first routed to the configured destination. In case the call cannot be placed, the service plays the second dial-tone or the configured announcement to the caller until the caller enters the first DTMF digit.

Caveat: For calls from the IP side, the service only works if the G711.alaw codec is chosen.

Mode: Context CS

Step	Command	Purpose
1	<code>node(ctx-cs)[switch]#service second-dialtone name</code>	Enters the service second-dialtone mode.
2	<code>node(svc-2dt)[name]#route call dest-interface dest-service dest-table name</code>	Defines the call destination. If the call cannot be placed the second-dialtone is played.
3	<code>node(svc-2dt)[name]#route announcement dest-interface dest-service dest-table name</code>	Instead of playing a dialtone locally, the service sets up a call to the configured message provider.
4	<code>node(svc-2dt)[name]#use profile tone-set name</code>	Apply to use a alternative tone-set profile. If not configured the profile tone-set default will be used.

Deleting Call Services

To remove individual call services you can use the **no** form of the **service** command.

Procedure: To delete a call service

Mode: Context CS

Step	Command	Purpose
1	<code>node(ctx-cs)[switch]#no service service-name</code>	Delete the service <i>service-name</i> . Note: You do not have to enter the type of the service when just deleting it. The type must only be specified when creating a service.

Example: Remove an entire mapping table

```
node(cfg)#context cs
node(ctx-cs)[switch]#no service HUNT-BRI
```

Activate the Call Router Configuration

Prior to activate the call router configuration you can show the whole context CS configuration and the entire call routing tables.

The call router configuration is activated as soon as the CS context comes out of shutdown (e.g. at boot time or by manually entering command **no shutdown**). You can modify the configuration at runtime; changes will be active after 3 seconds. Trinity offers a number of possibilities to monitor and debug the CS context and call router configurations. For more information refer to chapter 69, “[Debug and Monitoring](#)” on page 906.

Note It is not necessary to shutdown the CS context prior to making any configuration changes.

Procedure: To show and activate the call router configuration

Mode: Context CS

Step	Command	Purpose
1	<code>node(ctx-cs)[switch]#show call-router config</code>	Show the actual call router configuration. This displays all routing and mapping tables in the current context CS. When you are inside a routing or mapping table configuration mode, only the current table is displayed.
2	<code>node(ctx-cs)[switch]#show running-config</code>	Show the whole running config includes the call routing tables
3	<code>node(ctx-cs)[switch]#debug call-router detail level</code>	Enable the call router debug monitor. Use level 1 for get informed about errors and increase the level up to 5 to track calls during route lookups.
4	<code>node(ctx-cs)[switch]#no shutdown</code>	Activate the whole CS context configuration including the call router configuration.
5	<code>node(ctx-cs)[switch]#show call-router status</code>	Show the actual call router status. This command can be used to examine whether or not the call router accepted all routing entries as entered in the configuration.

Note You must explicitly enter the **no shutdown** command to activate the call router.

Test the Call Router Configuration

After activating the call router configuration you can test the call router by simulating a route lookup as if a call is routed to a table. You have to execute the test call-router command and specify all necessary call properties together with the routing table you want to test.

Note You must activate the call router using the **no shutdown** command first.

Procedure: To test the call router configuration

Mode: Context CS

Step	Command	Purpose
1	<code>node(ctx-cs)[switch]#debug cr</code>	Enables the call router debug monitor. Chose level 5 to trace route lookups in detail.
2	<code>node(ctx-cs)[switch]#test call-router table-name [property-type property-value]</code>	Tests the routing or mapping table <i>table-name</i> with the specified call property. You can repeat the optional section multiple times and thus enter as many call properties as you want.

Example: Create and test a routing table

```
node(cfg)#context cs
node(cts-cs)[switch]#routing-table called-e164 TEST
node(rt-tab)[TEST]#route 1 dest-interface IF1
node(rt-tab)[TEST]#route 1[0-4] dest-interface IF2
node(rt-tab)[TEST]#route 11 dest-interface IF3
node(rt-tab)[TEST]#route 111T dest-interface IF4
```

```

node(rt-tab)[TEST]#route default dest-interface IF5
node(rt-tab)[TEST]#exit
node(ctx-cs)[switch]#no shutdown
node(ctx-cs)[switch]#debug cr
node(ctx-cs)[switch]#test call-router TEST called-e164 123
Parameters
=====

Time:                2004-03-02T16:55:33<-- Time of the lookup
Result:              route-found-place-call<-- Lookup result
Destination:        IF2<-- Dest. Interface
Timeout:            0<-- Digit-Coll. TO

Property Containers
-----

Properties

Properties
  E164-Number:      123 (String)<-- CdPN after lookup/change

Properties

16:55:33 CR    > [switch] Routing-Lookup:
16:55:33 CR    >   Find best-matching called-element in table test
16:55:33 CR    >     01: Prefix Timeout Expression: E164-Number of 123 completely
(no timeout) matches 1
16:55:33 CR    >     02: Prefix Timeout Expression: E164-Number of 123 completely
(no timeout) matches 1[0-4]
16:55:33 CR    >     03: Prefix Timeout Expression: E164-Number of 123 does not
match 11
16:55:33 CR    >     04: Prefix Timeout Expression: E164-Number of 123 does not
match 111
16:55:33 CR    >       Selecting entry 2
16:55:33 CR    >       Execute all elements in table IF2
16:55:33 CR    >       Execute all elements in table route-found-place-call
16:55:33 CR    >       Lookup result: Route found; place call (timeout=0)

```

Example: Enterprise network with local breakout and IP carrier access

Consider the following Enterprise Network.

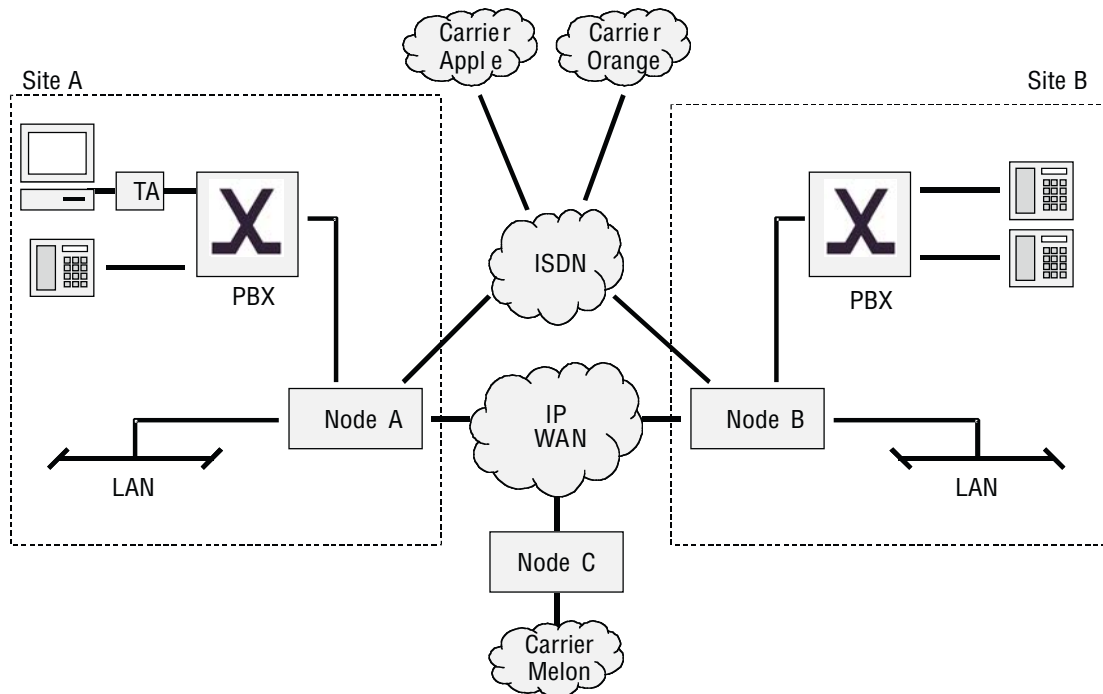


Figure 100. Call routing example network

Note The Patton devices in this Network may be owned and operated by the Company or by a Service Provider.

The goal of this scenario is to connect the two PBX of sites *A* and *B*. The two sites are connected to a broadband IP provider. The IP network is used to exchange data and voice calls between the two sites. On the IP network there is also a PSTN gateway (Node *C*) to an alternative voice carrier *Melon* that shall be used for most call destinations.

Sites *A* and *B* also have connections to the local ISDN network. This is called the local breakout connection. The local breakout is to be used as a fallback for ISDN data connections.

We assume the following:

- The number block for site *A* is 022 782 55 00 to 99
- The number block for site *B* is 033 665 2 000 to 999
- The Carrier Access Code (CAC) for *Apple* is 1055
- The Carrier Access Code (CAC) for *Orange* is 1066
- Carriers *Apple*, *Orange* and *Melon* do not support ISDN data calls (PC with ISDN Terminal Adapter behind PBX *A*)

- When calling through carrier *Melon* the CLI (calling party number) must not use the public number blocks of Site A and B
- Carrier *Orange* is to be used for national calls
- Carrier *Apple* is to be used for calls to mobile

The requirements for the call router can be summarized as:

1. Route ISDN data calls to the local breakout.
2. Route inter-site calls to the opposite Patton device (node A to node B and vice versa).
3. Route international calls to carrier *Melon*.
4. Provide a fallback for all VoIP calls on the local breakout.
5. Route local calls to the local breakout.
6. Route national calls to carrier *Orange*.
7. Route mobile calls to carrier *Apple*.
8. Calls from the PSTN, nodes B and C are forwarded directly to the PBX.

The remainder of this example will focus on the configuration for Node A. The configuration for Node B can be built accordingly. Node C has an even simpler configuration.

It is a good idea to specify the required call router elements and names before starting the configuration. A sketch may be helpful:

- Bearer capability table named *TAB-ISDN-SERVICE*, needed for requirement 1.
- Called party number table named *TAB-DEST-A*, needed for requirements 2, 3, 6 and 7
- CAC insertion for *Apple MAP-CAC-APPLE*, needed to add a carrier access code for *Apple*
- CAC insertion for *Orange MAP-CAC-ORANGE*, needed to add carrier access code for *Orange*
- CLI replacement for *Melon MAP-CLI-MELON*, needed to add carrier access code for *Melon*
- PSTN interfaces *IF-PBX-A* and *IF-LOCAL-BREAKOUT*, needed for requirements 4, 5 and 8.
- SIP interface *IF-NODE-C*, needed for requirement 3.

Figure 101 shows the corresponding CS Context and call router elements in node A:

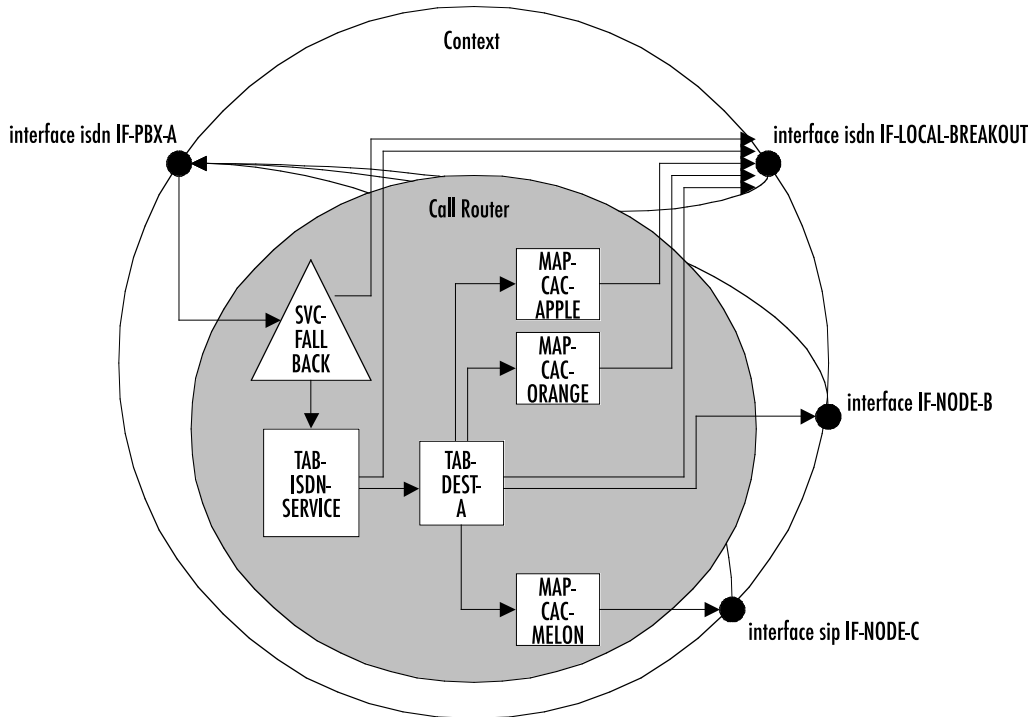


Figure 101. CS context and call router elements

We assume that the CS interfaces have already been created and configured. So we can start directly with the call router elements.

Since the command sequence is quite long it is useful to create the configuration offline and download it using TFTP.

Note In the following lines the prompt is omitted as in a configuration file and for better readability.

```
#-----
# Call Router Config File
#-----

context cs switch

#
# Hunt-group service "SVC-FALLBACK" to catch VoIP network errors
#

service hunt-group SVC-FALLBACK
  no cyclic
  timeout 6
  route call 1 dest-table ISDN-SERVICE
  route call 2 dest-interface LOCAL-BREAKOUT
```

```

#
# Bearer capability routing table "TAB-ISDN-SERVICE"
#

routing-table itc TAB-ISDN-SERVICE
  route unrestricted-digital dest-interface IF-LOCAL-BREAKOUT
  route default dest-table TAB-DEST-A

#
# Called party number routing table "TAB-DEST-A"
#

routing-table called-e164 TAB-DEST-A
  route 0 dest-interface IF-LOCAL-BREAKOUT MAP-CAC-ORANGE
  route 00 dest-interface IF-NODE-C MAP-CLI-MELON
  route 07[4-6] dest-interface IF-LOCAL-BREAKOUT MAP-CAC-APPLE
  route 0336652... dest-interface IF-NODE-B
  route default dest-interface IF-LOCAL-BREAKOUT

#
# Number manipulation "CAC-APPLE"; add prefix 1055
#

mapping-table called-e164 to called-e164 MAP-CAC-APPLE
  map (.%) to 1055\1

#
# Number manipulation "CAC-ORANGE"; add prefix 1066
#

mapping-table called-e164 to called-e164 MAP-CAC-ORANGE
  map (.%) to 1066\1

#
# Number manipulation "CLI-MELON"
# Truncate CLI to last 2 digits and add 08004455 prefix in front
#

mapping-table calling-e164 to calling-e164 MAP-CLI-MELON
  map .%(..) to 08004455\1

```

Prior to downloading this file you should make sure there are no other tables and functions in the call router.

```

node(ctx-cs)[switch]#copy tftp://172.16.36.20/configs/SRconf.cfg running-config
Download...100%

```

Now we have to enable advanced call routing for outgoing calls and basic interface routing for incoming calls: Calls arriving on the interface from the PBX are routed to the SVC-FALLBACK service while incoming calls from the other interfaces are routed directly to the IF-PBX-A interface.

```

node(ctx-cs)[switch]#interface isdn IF-PBX-A
node(if-pstn)[IF-PBX-A]#route dest-service SVC-FALLBACK
node(if-pstn)[IF-PBX-A]#exit
node(ctx-cs)[switch]#interface isdn IF-LOCAL-BREAKOUT
node(if-isdn)[IF-LOCA~]#route dest-interface IF-PBX-A
node(if-isdn)[IF-LOCA~]#exit

```

```
node(ctx-cs)[switch]#interface sip IF-NODE-C
node(IF-NODE-C)[IF-NODE-C]#route dest-interface IF-PBX-A
node(IF-NODE-C)[IF-NODE-C]#exit
```

The configuration is now complete. Prior to activating the configuration enable the call router debug monitor to check the loading of the call router elements.

```
node(cfg)#debug call-router
node(cfg)#context cs
node(ctx-cs)[switch]#no shutdown
02:14:30 CR    > Updating tables in 3 seconds...
02:14:33 CR    > [switch] Reloading tables now
02:14:33 CR    > [switch] Flushing all tables
02:14:33 CR    > [switch] Loading table 'TAB-ISDN-SERVICE'
02:14:33 CR    > [switch] Loading table 'TAB-DEST-A'
02:14:33 CR    > [switch] Loading table 'CAC-APPLE'
02:14:33 CR    > [switch] Loading table 'CAC-ORANGE'
02:14:33 CR    > [switch] Loading table 'CLI-MELON'
02:14:33 CR    > [switch] Loading table 'MAP-CAC-APPLE'
02:14:33 CR    > [switch] Loading table 'MAP-CAC-ORANGE'
02:14:33 CR    > [switch] Loading table 'MAP-CLI-MELON'
02:14:33 CR    > [switch] Loading table 'IF-LOCAL-BREAKOUT-precallservice'
02:14:33 CR    > [switch] Loading table 'IF-PBX-A-precallservice'
02:14:33 CR    > [switch] Loading table 'IF-NODE-B-precallservice'
02:14:33 CR    > [switch] Loading table 'IF-NODE-C-precallservice'
node(ctx-cs)[switch]#
```

Configuring Partial Rerouting

To save bandwidth and B-Channels, Trinity supports partial rerouting. Trinity accepts rerouting requests from PBX, which want to push-back a diverted or forwarded call with a Rerouting Request. Trinity can generate such a reroute request to push-back a looped call. Both, acceptance and emission of rerouting requests can be configured separately.

To prevent rerouting when a particular service is invoked in a call, services can be configured to allow or to forbid rerouting of calls of the service. In order that a call can be rerouted, all services participating in this call must allow rerouting. In addition rerouting emit must be configured on the interface where this call comes in and is routed out.

To accept or emit rerouting requests, see “[Call reroute](#)” on page 631.

To allow push-back, see “[Allow push-back](#)” on page 632.

Call reroute

The **call-reroute** commands enable acceptance and emission of rerouting requests.

Enable rerouting requests on ISDN. If a reroute is accepted, the participant who sends the reroute request is disconnected and the call is established from the Patton device to the new destination.

Mode: context cs/interface isdn

Step	Command	Purpose
1	<code>[name](if-isdn)[interface]#[no] call-reroute accept</code>	Enables acceptance of rerouting requests from ISDN. Default is disabled.

Enable emission of rerouting requests on ISDN. To reroute a call, you must enter and leave the device through the same ISDN interface and every service invoked must allow push-back.

Mode: context cs/interface isdn

Step	Command	Purpose
1	<code>[name](if-isdn)[interface]#[no] call-reroute emit</code>	Enables emission of rerouting requests from ISDN. Default is disabled.

Enable sending of “302 moved temporary” message on SIP. To reroute a call, you must enter and leave the device through the same SIP gateway and every service invoked must allow push-back.

Mode: context cs/interface sip

Step	Command	Purpose
1	<code>[name](if-sip)[interface]#[no] call-reroute emit</code>	Enables emission of “302 moved temporarily” message on SIP. Default is disabled.

Allow push-back

The **push-back** command allows or forbids rerouting of a call, if the service is invoked.

Enable push-back – aaa service.**Mode:** context cs/service aaa

Step	Command	Purpose
1	<i>[name]</i> (svc-aaa) <i>[service]</i> # [no] allows-push-back	Enables push-back of a call of this service. Default is disabled.

Enable push-back – bridge service.**Mode:** context cs/service bridge

Step	Command	Purpose
1	<i>[name]</i> (svc-brdg) <i>[service]</i> # [no] allows-push-back	Enables push-back of a call of this service. Default is disabled.

Enable push-back – distribution-group service.**Mode:** context cs/service distribution-group

Step	Command	Purpose
1	<i>[name]</i> (svc-dist) <i>[service]</i> # [no] allows-push-back	Enables push-back of a call of this service. Default is disabled.

Enable push-back – hunt group service.**Mode:** context cs/service hunt-group

Step	Command	Purpose
1	<i>[name]</i> (svc-hunt) <i>[service]</i> # [no] allows-push-back	Enables push-back of a call of this service. Default is disabled.

Enable push-back – limiter service.**Mode:** context cs/service limiter

Step	Command	Purpose
1	<i>[name]</i> (svc-lim) <i>[service]</i> # [no] allows-push-back	Enables push-back of a call of this service. Default is enabled.

Enable push-back – priority service.**Mode:** context cs/service priority

Step	Command	Purpose
1	<i>[name]</i> (svc-prio) <i>[service]</i> # [no] allows-push-back	Enables push-back of a call of this service. Default is enabled.

Chapter 50 **Global SIP Configuration**

Chapter contents

Introduction.....	635
Enabling SIP Client 'rport' Support	635
Configuring the SIP User-Agent Header	635
Configuring SIP Peer Flood Blocking.....	635
Limit Packets to Prevent SIP Overload Condition.....	636
Locking DNS Records for SIP Requests	636
SIP Request URI Length Limitation.....	637

Introduction

This chapter describes Trinity commands available under the global SIP configuration mode.

Enabling SIP Client ‘rport’ Support

Trinity supports the ‘rport’ parameter for SIP UAC according to RFC3581. This parameter forces a server that receives a request over UDP to send the response back to the IP-Address and Port from where it received the request and not to the Address/Port parameters available in the top most VIA header. This feature is useful for NAT-Traversal if the Patton device is located behind an external NAT.

This feature can only be enabled on a global base. All outgoing requests will contain the ‘rport’ parameter in the VIA header. The belonging command is available in the global sip configuration mode.

Mode: cfg

Step	Command	Purpose
1	[node](cfg)#sip	Enters the global SIP configuration mode
2	[node](sip)#[no] rport	Enables/disables the rport parameter in the VIA header of outgoing SIP requests. Default: disabled

Configuring the SIP User-Agent Header

The content of the SIP User-Agent header was hard-coded and not tailored to the Patton device.

The content of the SIP User-Agent header now pertains to the Patton device system information. Two different formats are supported, long and short. The long (default) format offers detailed information while the short format includes basic information. The format that is sent can be specified from the Web and CLI interfaces.

Mode: sip

Step	Command	Purpose
1	node(sip)# user-agent-header-format { long short }	Specifies the user-agent header format, default: long.

Note **Long format:** oem model serial-number sw-version hw-version.hw-revision
SIP-stack-version

Short format: oem model-up-to-the-first-slash serial-number sw-version

Configuring SIP Peer Flood Blocking

Trinity's SIP stack can be configured to prevent SIP packet flooding. This is useful to defend against SIP attacks of any sort where many SIP packets are being received from the same peer within a short period of time. Disabled by default, this feature can be enabled using a simple “switch”-style command, and fine-tuned with a separate configuration command.

The SIP Peer Flood Blocking feature works on a per-peer basis, and fixes a limit on the number of packets that the stack will receive from any given peer. That limit is based on a time window combined with a packet count

limit and a timeout. If, for any given peer, the number of packets received during the defined time window exceeds the limit, all further packets from that peer will be disregarded for the duration of the defined timeout.

The default parameters are dependent on the Patton device model, mostly on the CPU model and clock speed.

Mode: sip

Step	Command	Purpose
1	[node](sip)# [no] peer-flood-blocking	Enables/disables the Peer Flood Blocking feature

Configuration:

Step	Command	Purpose
1	node(sip)#config-peer-flood-blocking [<window-size> <packet-count-limit> <timeout>]	Configures the Peer Flood Block feature, or resets the parameters to the (product-specific) default values. This occurs when the parameters are omitted; however, if the parameters are used, all three parameters must be specified. Note The window-size and timeout values are in milliseconds.

Limit Packets to Prevent SIP Overload Condition

It is possible to limit the maximum amount of incoming SIP packets, which are stored to be handled and processed later. This guarantees a responsive system even in an overload condition. It handles and parses still as many requests as possible, but discards the excess packets.

When more SIP requests are arriving than Trinity can handle, the overall system performance decreases because the internal resources are occupied for requests which cannot be handled in time. This causes re-transmissions of requests and adds additional overhead to the system. Therefore, it is better to drop packets early in the processing queue in order to avoid a condition where no successful request processing is possible.

The best overall performance in processing the highest number of successful call attempts with a continuously overloaded system was reached by setting this queue limit to a value of 8.

Mode: sip

Step	Command	Purpose
1	[node](sip)#[no] max-queued-packets <number of packets>	Limits the internal SIP queue to discard any packets when the configured number of packets is reached. Suggested: 8

Locking DNS Records for SIP Requests

DNS A and SRV records need to be selected according to RFC3263 and RFC2782, therefore allowing load balancing. However, the defined algorithm is not desirable in all situations, e.g.:

- A SIP request (INVITE, REGISTER, SUBSCRIBE) is sent to a SIP server which requires authentication. The next SIP request containing authentication credentials is potentially sent to another server based the new DNS record list.
- A re-INVITE can also be sent to another server than the first INVITE.

In situations where this behavior is undesired, Trinity provides a global configuration option to lock the DNS record list. This option is enabled by default and affects all relevant SIP requests: INVITE, REGISTER and SUBSCRIBE.

Mode: sip

Step	Command	Purpose
1	[node](sip)[0/0]#[no] lock-dns-record	Enable locking of DNS records Default: enabled

SIP Request URI Length Limitation

This feature allows blocking incoming SIP request whose URI length exceeds a user specified value. Limiting the request URI length has device wide validity. The configured number of characters is applied for all incoming SIP requests on the whole device. Therefore the max-request-uri-length is located in the global sip configuration mode. By default the request URI length is unlimited.

Mode: sip

Step	Command	Purpose
1	node(sip)#[no] max-request-uri-length <length>	Enables/Disables request URI length limitation. The given length specifies the maximum number of characters including all URI parameters. Default: disabled

Chapter 51 SIP Overload Configuration

Chapter contents

Introduction	639
System Overview	639
Default Behavior	640
Mode of operation	640
SIP overload configuration task list.....	642
Disable SIP message rejection	642
Change the list of SIP messages rejected in an overload situation.....	643
Reset the SIP overload behavior to the system defaults	643
Display the behavior of SIP in an overload situation.....	644
Debug the behavior of SIP in an overload situation.....	646

Introduction

This chapter describes how to configure the device's behavior in case of an overload situation, in particular, how SIP calls shall be treated if the device is operated outside its nominal specification. In short, if the CPU load rises above a critical margin, new SIP calls are rejected. The same happens if the available RAM drops below a critical baseline.

Note The device automatically blocks new SIP calls in an overload situation by default and is pre-configured with optimal margins that define when the device is overloaded. Only experts should change this configuration. If you are unsure about whether or how to configure the SIP overload feature, please contact Patton's support team.

System Overview

As depicted in [figure 102](#), the behavior of SIP in an overload condition depends on a state profile, which itself is based on system variables and custom event expressions (see Chapter 11, “[Programmable System-Event Configuration](#)” on page 149).

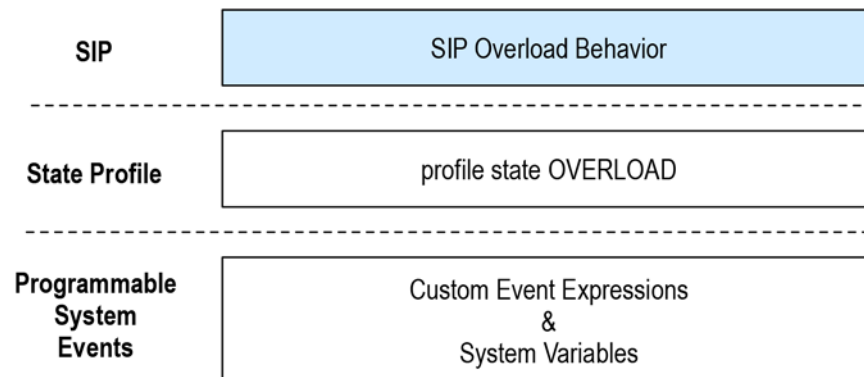


Figure 102. The SIP overload behavior uses the OVERLOAD state profile

From bottom to top these three blocks offer the following services:

- **Programmable System Events:** Trinity exposes many subsystem states as system variables (e.g. the link state of IP interfaces (`ip.ctx:ctx-name.if:if-name.up`), whether the time has been synchronized over NTP (`ntp.sync`), whether the system is up (`sys.up`), the available RAM (`sys.ram.avail`), etc.). The user may create custom expressions that combine system variables and other expressions with the `expression` command, e.g. `expression UPNSYNC "sys.up && ntp.sync"`. Whenever a system variable or custom expression changes its value a notification event is sent to all interesting applications, e.g. to the state profiles.
- **State Profile:** A state profile builds a simple state machine from system variables and custom expressions. Change-notification events from those variables are used to describe the transitions between its states. The device contains a built-in state profile called `OVERLOAD`. This overload state machine is in state `NORMAL` if the device operates according its nominal specification. The state machine transitions to the `WARNING` state if the CPU utilization was pretty high for a minute (**default 80%**)The state machine further transitions to the `CRITICAL` state if the CPU utilization was almost 100% for a minute (**default 95%**).

- **SIP Overload Behavior:** On top of these two components the SIP overload behavior links to a state profile (to the *OVERLOAD* profile by default) and defines the behavior of the SIP user agent in each state (*NORMAL*, *WARNING*, and *CRITICAL*). By default new SIP calls are rejected in the *WARNING* state whereas most SIP requests are rejected in the *CRITICAL* state. Only requests to close a call are passed in the latter state. See [table 34](#) for more details.

This chapter explains how to configure the SIP overload behavior based on an existing state profile. See Chapter 51, “[SIP Overload Configuration](#)” on page 638 for information on how to change the conditions for the state transitions or how to create a new state profile.

Default Behavior

The following table lists the default behavior of the SIP user agent in any of the overload states:

Table 34. Default SIP overload behavior

State	Behavior of the SIP user agent
NORMAL	The Patton device accepts all incoming SIP messages and places all outgoing SIP calls.
WARNING	The Patton device rejects all incoming INVITE, OPTIONS, PING, REGISTER, and SUBSCRIBE requests by sending back a 503 Server Out Of Resources response. The Patton device also rejects all incoming responses to previous INVITE, OPTIONS, PING, REGISTER, and SUBSCRIBE requests sent by the Patton device. The Patton device does not place outgoing SIP calls anymore.
CRITICAL	The Patton device rejects all incoming SIP requests. Only requests required to drop a call are passed: ACK, BYE, CANCEL . The Patton device rejects all incoming SIP responses. Only incoming responses to the following requests are still passed to the user agent application: ACK, BYE, CANCEL . As in the WARNING state the Patton device does not place outgoing SIP calls anymore.

If the default configuration does not suit your needs please follow the procedure outlined in the next section to change the overload behavior.

Rejecting SIP messages might look strange. In an overload situation however, if the device is operated outside its nominal specification, rejecting SIP messages makes sure that the following properties are retained:

To reject calls if the CPU load is too high allows the operator to still re-configure and monitor the device if the entire CPU capacity is used for other tasks (e.g. handling SIP calls). Each SIP message requires some CPU power to be processed. If the rate of processed SIP request becomes too high, the SIP user agent would consume all CPU power and the user interfaces (CLI, WebUI) would be slowed down drastically. By rejecting SIP calls if the CPU utilization is too high the device stays responsive to management operations.

Mode of operation

There are two hooks in the SIP user agent that filter all incoming messages and outgoing calls (see [figure 103](#) on page 641). All incoming SIP messages have to pass the *inbound SIP message filter* before they reach the SIP gateway (user agent). All outbound SIP calls routed by the call-router to a SIP interface have to pass the *outbound SIP call filter*. Whether or not those two filters permit or deny the message/call to pass is configured in

the sip / overload CLI configuration mode. If you deny INVITE requests, incoming INVITE messages as well as outgoing calls are rejected. If you deny SIP requests other than INVITE, only those received requests are affected (no outbound message filter is in place).

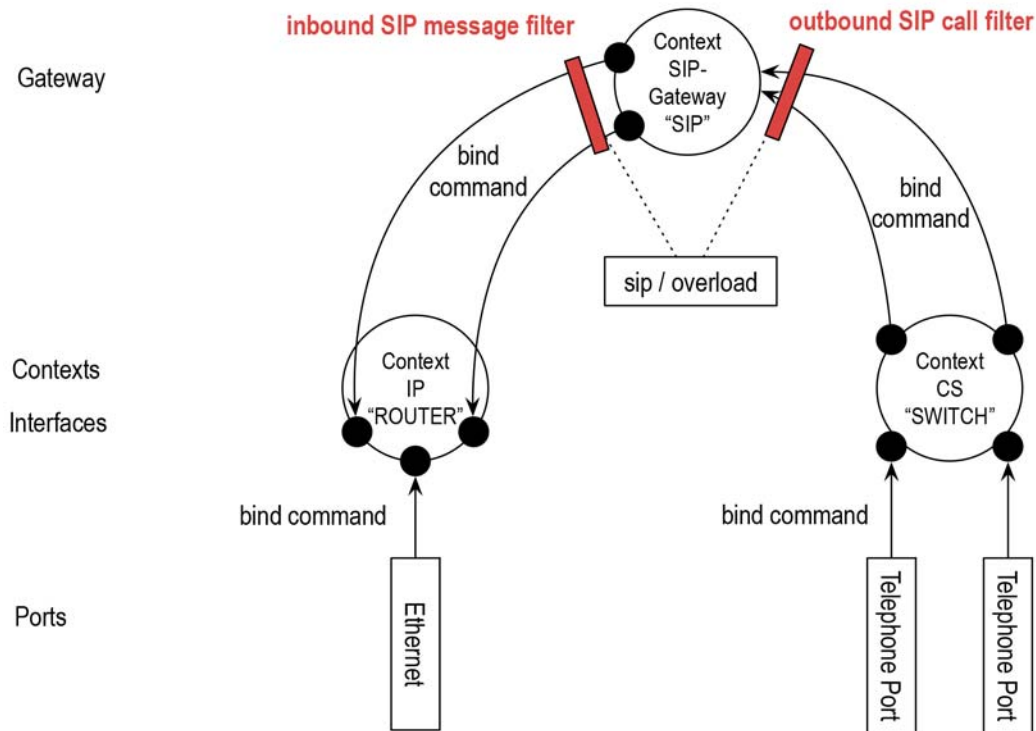


Figure 103. SIP Overload configuration and message flow diagram

Figure 104 on page 642 depicts in detail what happens to an incoming SIP request or an outgoing SIP call when it reaches the inbound or outbound SIP message or call filter, respectively. The SIP overload configuration consists of a set of Access Control Lists (ACL) for SIP methods (INVITE, BYE, etc.). There is a separate SIP ACL for every state of the used state profile. Thus if you are using the default *OVERLOAD* state profile there are separate SIP ACLs for the three states *NORMAL*, *WARNING*, and *CRITICAL*. A SIP ACL consists of a set of rules each describing whether a set of SIP methods shall be permitted (passed) or denied (rejected).

A SIP message is handed over to the ACL that corresponds to the current state of the used state profile. For example if the device is in *WARNING* state, the first rule rejects the message if it is an INVITE, OPTION, PING, REGISTER, or SUBSCRIBE request or a response to such a request. Since this is the only rule in the *WARNING*-ACL the message is permitted (passed to the user agent). Actually, all message not explicitly mentioned in the SIP ACL are permitted. If you want to deny all un-mentioned messages in a state you have to explicitly permit some messages and append a deny-all rule at the end (see the SIP ACL for the *CRITICAL* state).

Note The SIP ACL works in a similar way than the Access Control List for data packets (see Chapter 38, “[Access Control List Configuration](#)” on page 440) but solely operates on incoming SIP packets after they have passed the data ACL.

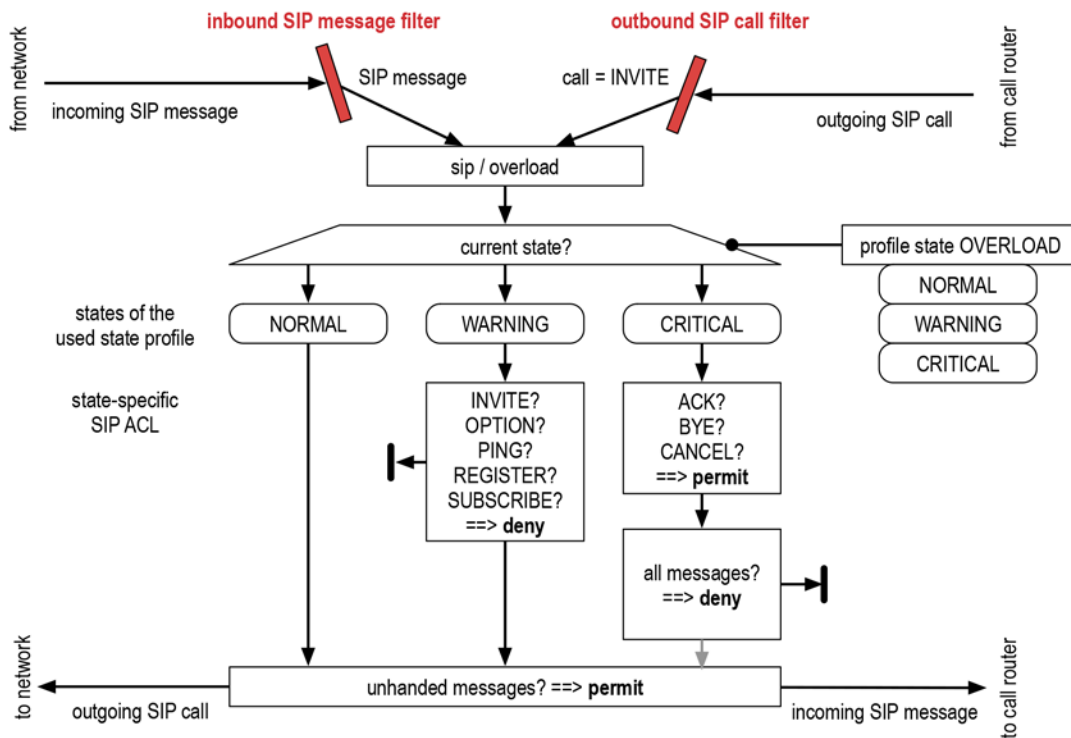


Figure 104. State-Based SIP Access Control Lists (ACL)

SIP overload configuration task list

To change the default behavior of the SIP user agent in an overload condition, perform one of the tasks in the following sections.

- “Disable SIP message rejection”
- “Change the list of SIP messages rejected in an overload situation” on page 643
- “Reset the SIP overload behavior to the system defaults” on page 643
- “Display the behavior of SIP in an overload situation” on page 644
- “Debug the behavior of SIP in an overload situation” on page 646

Disable SIP message rejection

As stated above rejecting SIP messages is a useful approach to make sure that the device is still responsive if operated outside its nominal specification. If you want to disable SIP message blocking nevertheless please follow the procedure below:

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg)#sip</code>	Enter the global SIP configuration mode.
2	<code>node(sip)#overload</code>	Enter the SIP overload configuration mode.

Step	Command	Purpose
3	<code>node(overload)#no use profile state</code>	Enter the SIP overload configuration mode.

The use *[no] profile state* command enables or disables the SIP overload feature; there is no explicit shutdown command.

Change the list of SIP messages rejected in an overload situation

The following procedure shows how to modify the SIP ACL that is passed by SIP messages in a specific state. Steps 4 and 5 may be repeated to add other states or rules in those states.

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg)#sip</code>	Enter the global SIP configuration mode.
2	<code>node(sip)#overload</code>	Enter the SIP overload configuration mode.
3	<code>node(overload)#use profile state name</code>	Link the SIP user agent to a state profile that defines the overload condition. From now on all incoming SIP messages are filtered by the ACL configured below. By default, the <i>OVERLOAD</i> state profile is used.
4	<code>node(overload)#state name</code>	Enter the ACL mode of a specific state. The <i>name</i> parameter must match the name of a state defined by the state profile linked in step 3.
5	<code>node(state)[name]#(permit deny) (all sip-method*)</code>	Append a SIP ACL rule for the current state, which is to permit or deny packets of the listed <i>sip-methods</i> or of all messages.

Example: Drop all but INVITE messages in the CRITICAL state.

```
node>enable
node#configure
node(cfg)#sip
node(sip)#overload
node(overload)#use profile state OVERLOAD
node(overload)#state CRITICAL
node(overload)#no deny 2 # remove existing rule
node(overload)#no permit 1 # remove existing rule
node(overload)#permit INVITE
node(overload)#deny all
```

Reset the SIP overload behavior to the system defaults

If you have experimented with either changing the conditions of the OVERLOAD state profile or changed the SIP ACLs in the sip/overload configuration mode, the following procedure re-establishes the system default behavior.

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg)#reset profile state OVERLOAD</code>	Reset the <i>OVERLOAD</i> state profile to its default states and conditions.
2	<code>node(cfg)#sip</code>	Enter the global SIP configuration mode.
3	<code>node(sip)#reset overload</code>	Reset the SIP overload behavior (linked profile and SIP ACLs for all states) to its default configuration.

Example: Reset both the default *OVERLOAD* state profile and the SIP overload behavior to the system-default configuration.

```
node>enable
node#configure
node(cfg)#reset profile state OVERLOAD
node(cfg)#sip
node(sip)#reset overload
```

Display the behavior of SIP in an overload situation

The following CLI command displays the configuration and statistics of the SIP overload feature:

Mode: Operator exec

Step	Command	Purpose
1	<code>node>show sip overload</code>	Displays the linked state profile, the current state and an action table for each SIP message, whether the message will be permitted or denied in each state.

Example: Display information about the current SIP overload behavior

```
node>show sip overload
SIP Overload Behavior
=====

State Profile:                OVERLOAD
Current State:                NORMAL
Status:                       enabled

Behavior
-----

State: CRITICAL
-----

Configured Rules
-----
  permit: ACK, BYE, CANCEL
  deny:   ACK, BYE, CANCEL, INFO, INVITE, MESSAGE, NOTIFY,
         OPTIONS, PING, PRACK, PUBLISH, REFER, REGISTER,
         SERVICE, SUBSCRIBE, UPDATE

Compiled Action Table
```



```

-----
ACK:                permit
BYE:                permit
CANCEL:            permit
INFO:              deny
INVITE:            deny
MESSAGE:           deny
NOTIFY:            deny
OPTIONS:           deny
PING:              deny
PRACK:             deny
PUBLISH:           deny
REFER:             deny
REGISTER:          deny
SERVICE:          deny
SUBSCRIBE:         deny
UPDATE:           deny

State: WARNING
-----

Configured Rules
-----
deny: INVITE, OPTIONS, PING, REGISTER, SUBSCRIBE

Compiled Action Table
-----
ACK:                permit
BYE:                permit
CANCEL:            permit
INFO:              permit
INVITE:            deny
MESSAGE:           permit
NOTIFY:            permit
OPTIONS:           deny
PING:              deny
PRACK:             permit
PUBLISH:           permit
REFER:             permit
REGISTER:          deny
SERVICE:          permit
SUBSCRIBE:         deny
UPDATE:           permit

Statistics
-----

Incoming SIP Requests
-----
ACK:    Rx:  35 | Permitted:  35 | Denied: .
CANCEL: Rx:  36 | Permitted:  36 | Denied: .
INVITE: Rx:  66 | Permitted:  66 | Denied: .

Incoming SIP Responses
-----

```

```

CANCEL: Rx: 35 | Permitted: 35 | Denied: .
INVITE: Rx: 160 | Permitted: 160 | Denied: .

Outgoing SIP Requests
-----
INVITE: Rx: 64 | Permitted: 64 | Denied: .

```

The first section shows the linked *State Profile* and the *Current State* of the linked state machine.

The second section lists all states mentioned in the SIP overload configuration. For each state the configured SIP ACL rules are printed. In addition the show command displays the compiled message table that assigns each SIP request an action (permit or deny).

Finally, the third section displays statistics about how many messages have been received and how many of those have been permitted or denied. *Incoming SIP Requests* refer to request messages filtered at the inbound SIP message filter hook (see [figure 104](#) on page 642) whereas *Incoming SIP Responses* are responses received at the same hook. *Outgoing SIP Requests* refer to outbound SIP calls received from the call-router that are handled on the outbound SIP call filter.

Note If the SIP overload feature is disabled (*no profile state* command has been entered) the SIP packets are not counted anymore.

Debug the behavior of SIP in an overload situation

Each SIP message that is analyzed by one of the SIP message filters is logged to the sip-signaling monitor (detail level 3).

Mode: Operator exec

Step	Command	Purpose
1	<code>node>debug call-signaling [detail level full-detail]</code>	Enables the real-time debugger that prints information about SIP call-signaling events. The SIP message filters will log to this monitor when permitting or denying a SIP message.

Example: Observe how an incoming SIP call is rejected in an overload situation (state CRITICAL).

```

node>debug call-signaling detail 3
18:47:00.000 SIP-SIG # [Overload] Received INVITE request in state CRITICAL
18:47:00.000 SIP-SIG # [Overload] deny SIP packet
18:47:00.000 SIP-SIG # [Overload] Received ACK request in state CRITICAL
18:47:00.000 SIP-SIG # [Overload] permit SIP packet
18:47:00.000 SIP-SIG # [STACK] UNMATCHED ACK REQUEST

```

Note The *UNMATCHED ACK REQUEST* message indicates that the SIP stack received an ACK message without receiving an INVITE message before. This is because we denied INVITE requests but permitted ACK requests, a behavior required for properly terminating calls in an overload situation.

Chapter 52 SIP Interface Configuration

Chapter contents

Introduction	648
SIP Interface Configuration Task List	649
Binding the Interface to a SIP Gateway	649
Configuring a Remote Host	650
Managing trusted hosts	650
Configuring a Local Host (Optional)	650
Using an Alternate VoIP Profile (Optional)	651
Using an Alternate SIP Profile (Optional)	651
Using an Alternate Tone-Set Profile (Optional)	652
Configuring Early Call Connect/Disconnect (Optional)	652
Enable/Disable Early-proceeding on SIP Interface	653
Early Media Behavior	653
Configuring Address Translation (Optional)	653
Mapping call-control properties in SIP headers	653
Mapping SIP headers to call-control properties	654
Configuring ISDN redirecting number tunneling over SIP	654
Enabling SIP RFC privacy, asserted-identity, & preferred-identity headers (RFC 3323/3325)	655
Updating caller address parameters	655
SIP diversion header	656
Transmit Direction	656
Receive Direction	657
SIP Calling Party Number Handling	657
SIP REFER Transmission (& ISDN Explicit Call Transfer support)	657
AOC Over SIP (Optional)	658
Enabling the Session Timer (Optional)	659
Enabling the SIP Penalty-box Feature (Optional)	660
Initiating a New SIP Session for Redirected SIP Calls (Optional)	661
Rerouting Calls from SIP (Optional)	661
Configuring the SIP Hold Method (Optional)	661
Configuring SIP Overlap Dialing (Optional)	661
Configuring PRACK for Reliable Provisional Responses (optional)	662
Configuring History-Info in SIP	663
Enabling NAT Traversal for SIP INVITE Messages	664
Accepting Re-invite After Reboot	664

Introduction

This chapter provides an overview of SIP interfaces used by context SIP gateways and describes the specific tasks involved in their configuration. This chapter does not explain the basic configuration steps required for all CS interfaces. See Chapter 45, “CS Interface Configuration” on page 521 for information about configuring basic CS interfaces.

Within the CS context, a SIP interface is a special type of CS interface providing call routing for incoming and outgoing calls to and from the context SIP gateway (see [figure 105](#)).

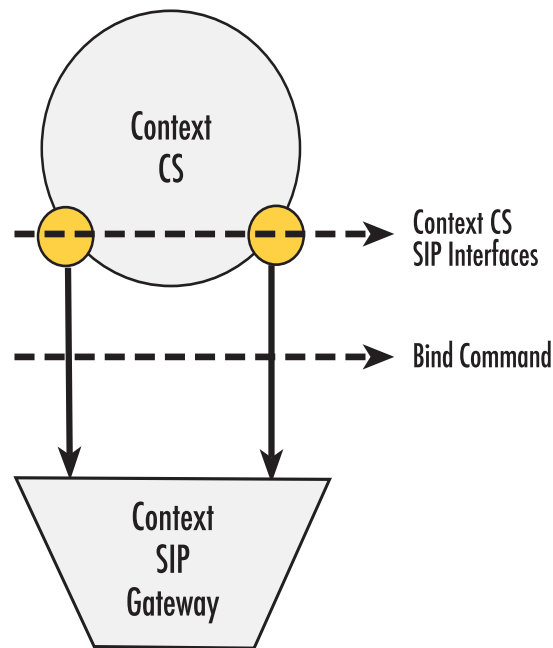


Figure 105. SIP interfaces on the CS context

A SIP interface is a CS interface type that is responsible for the address translation between the SIP environment and the call control. It provides configurable translation rules that define which parameter of which SIP header has to be translated to call control properties and vice versa. In addition, it offers VoIP settings and the possibility to configure SIP supplementary services like Call Transfer, Call Reroute or Session Timer. All SIP interfaces must be explicitly bound to a context SIP gateway. Calls that are routed from the Context CS to one of the SIP interfaces will be forwarded for call establishment to the context SIP gateway to which the SIP interface is bound. All of the parameters configured in the SIP interface will be applied to the forwarded call.

SIP Interface Configuration Task List

This section describes the configuration tasks for SIP interfaces listed below. You must at least perform the tasks that are not marked as optional to define a working SIP interface. The optional tasks are only required in advanced configurations. Before you can start with the SIP interface specific configuration tasks, you need to create the SIP interface and define the routing for it as defined in Chapter 45, “CS Interface Configuration” on page 521.

- Binding the interface to a context SIP gateway (see [page 649](#))
- Configure a remote host (see [page 650](#))
- Configure a local host (Optional) (see [page 650](#))
- Using an alternate VoIP profile (Optional) (see [page 651](#))
- Using an alternate SIP profile (Optional) (see [page 651](#))
- Using an alternate Tone-Set profile (Optional) (see [page 652](#))
- Configuring early call connect /disconnect (Optional) (see [page 652](#))
- Early Media Behavior (see [page 653](#))
- Configuring address translation (Optional) (see [page 653](#))
- SIP Calling Party Number Handling (see [page 657](#))
- SIP REFER Transmission (& ISDN Explicit Call Transfer support) (Optional) (see [page 657](#))
- AOC over SIP (Optional) (see [page 658](#))
- Enable the session timer (Optional) (see [page 659](#))
- Enable the SIP penalty-box feature (Optional) (see [page 660](#))
- Initiating a new SIP session for redirected SIP calls (Optional) (see [page 661](#))
- Configure the SIP hold method (Optional) (see [page 661](#))
- Configuring PRACK for Reliable Provisional Responses (optional)
- Configuring History-Info in SIP (see [page 663](#))
- Enabling NAT Traversal for SIP INVITE Messages (see [page 664](#))
- Accepting Re-invite After Reboot (see [page 664](#))

Binding the Interface to a SIP Gateway

Every SIP interface must be explicitly bound to a context SIP gateway. It defines the transport interfaces to be used to send requests and also from where it expects to receive incoming calls.

Mode: Interface SIP

Step	Command	Purpose
1	[node](if-sip)[if-name]# [no] bind context sip-gateway gw-name	Binds the interface to a context SIP gateway. The no form of the command removes an existing binding. Default: none

Configuring a Remote Host

The remote host parameter is used to build the host part of the To-Header-URI and the Request-URI. All calls forwarded to the context SIP gateway through this SIP interface will be sent to that host unless an outbound proxy has been configured for the outgoing Request-URI. In this case, all requests will be sent to the specified proxy independent of the Request-URI. For details about proxy configuration, see Chapter 48, “[Location Service](#)” on page 540.

The remote host parameter may contain a Domain Name, a Full Qualified Domain Name or an IP Address. If domain names are used, do not forget to configure the DNS client for address resolving. See Chapter 32, “[DNS Configuration](#)” on page 386 for more information.

An optional port number can be entered on this command. It specifies the destination port of the requests to be sent. If no port has been defined, the default SIP signaling port 5060 will be taken.

Mode: Interface SIP

Step	Command	Purpose
1	[node](if-sip)[if-name]# [no] remote <hostname> [port]	Specifies the remote host name and port. The no form of the command removes a configured host name. Default Hostname: none Default Port: 5060

Managing trusted hosts

A list of trusted remote peers can be configured on SIP interfaces. If configured, only connections with peers in that list will be accepted. The list may contain IP-addresses or FQDNs or the keyword remote. If no trusted peers are configured all connections are accepted.

Mode: Interface SIP

Step	Command	Purpose
1	[node](if-sip)[if-name]# [no] trust { remote <ip-address> <domain-name> }	Adds/removes a host to/from the trusted host list. Enter either an ip-address, a fully qualified domain name or the keyword <i>remote</i> . The <i>remote</i> keyword will add whatever is configured in the SIP interface’s remote command.

Configuring a Local Host (Optional)

The local host parameter is used to build the host part of the From-Header-URI. Optionally, a port can be entered. If no port has been specified, it will not appear in the From-Header-URI. This command is optional

in a SIP network setup where no DNS names are involved and only IP addresses applied. If no local host name has been specified, the IP address of the outgoing IP interface will be taken as host part of the From-Header-URI.

There are some exceptions where a local host name must be entered. One exception is if the bound gateway has two transport interfaces and the From-Header-URI call outbound properties have been configured. Such a call outbound property is the outbound proxy. To find the right identity in the location service, the From-Header-URI must exist. But, if there is more than one transport interface, it is not clear which one will be used to send the request because the proxy influences the routing. In such a case, it is recommended that the user configures the local host name and knows which IP interface the request should be sent over.

The SIP server expects its own IP address in the From-Header-URI because of security or routing reasons. This can be solved by manually entering a local host name.

Mode: Interface SIP

Step	Command	Purpose
1	[node](if-sip)[if-name]# [no] local <i><hostname></i> [port]	Specifies the local host name and port. The no form of the command removes a configured host name. Default Hostname: none Default Port: none

Using an Alternate VoIP Profile (Optional)

A VoIP profile is a container for all data path-related settings on VoIP connections. The predefined default profile exists persistently in the system and it is preconfigured with proper default parameters. It will always be taken if no other profile has been specified. This command allows the user to specify an alternate VoIP profile to use for all calls routed over this interface. For details about VoIP profile configuration see Chapter 42, “VoIP Profile Configuration” on page 470.

When a profile has been specified on the interface, it is possible to provide a special one for specific identities or group of identities. It can be configured in location service identities for call inbound and call outbound. See Chapter 48, “Location Service” on page 540 for details about identities and VoIP profile configurations. The identity lookup to find a possible configured VoIP profile will always be applied to the remote SIP URI. For outgoing calls, the SIP Request-URI will be taken. For incoming calls, the SIP From-URI will be taken.

Mode: Interface SIP

Step	Command	Purpose
1	[node](if-sip)[if-name]#use profile voip <i><profile-name></i>	Defines an alternate VoIP profile to be used for this SIP interface. Default: default

Using an Alternate SIP Profile (Optional)

The SIP profile contains a cause and reason mapping from SIP notation to the call control notation and vice versa. The predefined default profile exists persistently in the system and it is preconfigured with proper default mappings. It will always be taken if no other profile has been specified. This command allows the user to specify an alternate SIP profile to use for all calls routed over this interface. For details about SIP profile configuration see Chapter 41, “SIP Profile Configuration” on page 464.

When a profile has been specified on the interface, it is possible to provide a special one for specific identities or group of identities. It can be configured in location service identities for call inbound and call outbound. See Chapter 48, “[Location Service](#)” on page 540 for details about identities and SIP profile configurations. The identity lookup to find a possible configured SIP profile will always be applied to the remote SIP URI. For outgoing calls, the SIP Request-URI will be taken. For incoming calls, the SIP From-URI will be taken.

Mode: Interface SIP

Step	Command	Purpose
1	[node](if-sip)[if-name]#use profile sip <i><profile-name></i>	Defines an alternate SIP profile to be used for this SIP interface. Default: default

Using an Alternate Tone-Set Profile (Optional)

A Tone-Set profile contains a mapping of call-progress tones to defined tone sequences. The predefined default profile exists persistently in the system and it is preconfigured with the Swiss standard call-progress tone mapping. It will always be taken if no other profile has been specified. This command allows the user to specify an alternate Tone-Set profile to be use for all calls routed over this interface. For details about Tone-Set profile configuration see Chapter 46, “[Tone Configuration](#)” on page 530.

When a profile has been specified on the interface, it is possible to provide a special one for specific identities or group of identities. It can be configured in location service identities for call inbound and call outbound. See Chapter 48, “[Location Service](#)” on page 540 for details about identities and Tone-Set profile configurations. The identity lookup to find a possible configured Tone-Set profile will always be applied to the remote SIP URI. For outgoing calls, the SIP Request-URI will be taken. For incoming calls, the SIP From-URI will be taken.

Mode: Interface SIP

Step	Command	Purpose
1	[node](if-sip)[if-name]#use profile tone-set <i><profile-name></i>	Defines an alternate Tone-Set profile to be used for this SIP interface. Default: default

Configuring Early Call Connect/Disconnect (Optional)

Normally, SIP calls are fully connected by sending a 200 OK response to the INVITE request, if the called party answers the call. Any call progress tones or announcements are transmitted in the early SIP dialog. However, there are several SIP user agents that do not support media to be transmitted or received in an early dialog. To solve this problem, it is possible to connect the SIP call using a 200 OK response to the initial INVITE request as soon as inbound information is available. This will allow any SIP user agent to receive precall inbound information.

Early call disconnect suppresses busy tones (e.g. disturbing a telephone conference) and post-call announcements by sending a BYE message to the remote SIP user agent when the connected terminal hangs up (ISDN: when Disconnect message is received; analog line: when busy tone is detected, loop current is interrupted, or battery voltage is reversed).

Mode: Interface SIP

Step	Command	Purpose
1	[node](if-sip)[if-name]#[no] early-connect	Enables/Disables early call connect Default: disabled
2	[node](if-sip)[if-name]#[no] early-disconnect	Enables/Disables early call disconnect Default: disabled

Enable/Disable Early-proceeding on SIP Interface

In SBC scenarios, it is possible that multiple provisional responses (1xx) are not correctly forwarded. This can be corrected by *disabling* the early-proceeding parameter on SIP interfaces. In case of PSTN gateway scenarios, it is preferable to leave the early-proceeding parameter set to *enabled* on SIP interfaces.

Mode: Interface SIP

Step	Command	Purpose
1	[node](if-sip)[if-name]#[no] early-proceeding	Switches to proceeding state without receiving any provisional response. Default: enabled

Early Media Behavior

This command allows controlling SDP announcements in provisional SIP responses. It is possible to prevent any announcement (no early-media emit) or to instruct the system to do it automatically. In auto mode, SDP announcement is done based on call properties.

Mode: Interface SIP

Step	Command	Purpose
1	device(if-sip)[name]#[no] early-media emit {auto}	Defines SDP announcement procedure for provisional responses. Default: auto

Configuring Address Translation (Optional)

Mapping call-control properties in SIP headers

This functionality specifies rules for building the SIP headers for outgoing SIP requests. The user can configure which call-control property should take place as the user-part, the host-part or as additional parameter in a SIP header's URI.

Mode: Interface SIP

Step	Command	Purpose
1	[node](if-sip)[if-name]#[no] address-translation outgoing-call {header} {parameter} {source} {property}	Specifies an outgoing address translation rule.

Mapping SIP headers to call-control properties

This functionality specifies rules that describe which SIP header should take place as a specific call-control property like the called-e164 or the calling-e164 number.

Mode: Interface SIP

Step	Command	Purpose
1	[node](if-sip)[if-name]#[no] address-translation incoming-call {property} {header}	Specifies rule to extract call-control <i>property</i> from a <i>header</i> in the incoming call.
2	[node](if-sip)[if-name]#[no] address-translation incoming-call called-e164 { fix request-uri to-header p-called-party-id }	Specifies rule to extract <i>called-e164</i> number from one of the headers: fix, request-uri, to-header or p-called-party-id (Default header: request-uri) in the incoming call.
3	[node](if-sip)[if-name]#[no] address-translation incoming-call called-uri { fix local local-ip request-uri to-header p-called-party-id }	Specifies rule to extract <i>called-uri</i> from one of the headers: fix, local, local-ip, request-uri, to-header or p-called-party-id (Default header: request-uri) in the incoming call.
4	[node](if-sip)[if-name]#[no] address-translation incoming-call called-name { fix to-header p-called-party-id }	Specifies rule to extract <i>called-name</i> property from one of the headers: fix, to-header or p-called-party-id (Default header: to-header) in the incoming call.

Configuring ISDN redirecting number tunneling over SIP

A SIP interface can be configured to tunnel the ISDN Redirecting E.164 Number and Redirecting Reason. This is implemented per the IETF draft standard method according to draft-jennings-sip-voicemail-uri-05.

Mode: Interface SIP

Step	Command	Purpose
1	[node](if-sip)[if-name]# address-translation outgoing-call	Enables Redirecting Party Number Tunneling ISDN ‡ SIP: Enables transmission of the target and cause parameters in the Request-URI for outgoing SIP calls. Whenever the (incoming ISDN) call has a redirecting party number information element, the Request-URI is extended by the target and cause parameters. Default: disabled

Step	Command	Purpose
2	[node](if-sip)[if-name]# address-translation incoming-call	Enables Redirecting Party Number Tunneling SIP ‡ ISDN: Enables reception of the target and cause parameters in the Request-URI for incoming SIP calls. Sets the redirecting party number information element for (outgoing ISDN) calls where the target parameter is present in the Request-URI. Default: disabled

Enabling SIP RFC privacy, asserted-identity, & preferred-identity headers (RFC 3323/3325)

The following command sequence enables support for the SIP Privacy and Asserted-Identity headers, sending and receiving of the Privacy as well as the Asserted-Identity headers of RFCs 3323 & 3325. This provides the required identity to SIP entities. The privacy header suppresses the forwarding of the identity to the final station. Handling of the Asserted-Identity header can be configured in the same way as for any other SIP header using the address-translation command in the SIP interface configuration mode. The privacy header is mapped to the call-control's presentation indicator property field.

The type of header sent depends on the screening-indicator property of the call-control. When receiving one of these privacy headers the screening-indicator is also set depending on the received header type.

Mode: Interface SIP

Step	Command	Purpose
1	[node](if-sip)[if-name]#[no] privacy	Enables/Disables privacy. Default: disabled

Updating caller address parameters

A SIP User Agent Client (UAC) can use the sip update method for modifying caller-name and caller-number. These parameters must be included in the P-Asserted-Identity header of the sip update message. This feature is often required if sip has to interwork with an analog telephone network where the Caller-Id cannot be delivered together with the call offering.

It is possible to configure the sip interface to wait for the caller-name, caller-number or both parameters before routing the call. If the caller-address parameters are sent with sip update and the call will be forwarded to a network where these parameters must be present at call setup time (ISDN), this wait-service must be enabled.

Note The SIP interface of the call-router must be configured to take the caller addresses parameters for incoming calls from the P-Asserted-Identity header. See “[Enabling SIP RFC privacy, asserted-identity, & preferred-identity headers \(RFC 3323/3325\)](#)” on page 655 for more information.

Mode: Interface SIP

Step	Command	Purpose
1	<code>[node](if-sip)[interface]#[no] update accept address {wait-for-name wait-for-number wait-for-name-and-number} [[proceeding-timeout <timeout>][early-alerting timeout <timeout>]]</code>	<p>Enables or disables wait service:</p> <p>Wait Options:</p> <ul style="list-style-type: none"> • wait-for-name The call will be routed as soon as a valid caller name is present. • wait-for-number The call will be routed as soon as a valid caller number is present. • wait-for-name-and-number The call will be routed as soon as a valid caller name and a valid caller number are present. <p>Wait Timeouts:</p> <ul style="list-style-type: none"> • proceeding-timeout After this time, the call will be routed and the specified caller parameters are not present. <i>Default:</i> 4000ms • early-alerting-timeout If the specified caller parameters are not present after this time, a 180 RINGING will be sent to UAC. <i>Default:</i> 0ms (immediately)

SIP diversion header

Trinity supports the SIP Diversion Header for transmitting redirecting information over SIP according to draft-levy-sip-diversion-08. Sending and receiving of the header can be configured independent of each other. Even though the Diversion Header standard would allow appending a header for each diversion occurred in the network, Trinity only records the last and the first diversion. If only one Diversion Header is attached to the INVITE request, then it represents the last diversion.

Transmit Direction. For enabling sending of the Diversion Header, an outgoing address translation expression must be configured on the sip interface. This expression specifies how to create the Diversion URI of the header. As the User Part of the URI, the Calling Redirecting number will always be taken. The user must configure the Host Part that is set per default to none. Setting the Host Part to none disables transmission of the Diversion Header.

Mode: Interface SIP

Step	Command	Purpose
1	<code>[name] (if-sip)[interface]#address-translation outgoing-call diversion-header host-part {call local remote fix interface none}</code>	Enables or disables sending of the Diversion Header and specifies the Host Part of the URI.

Receive Direction. For receiving of the Diversion Header, an incoming address translation expression must be configured on the sip interface. Because several methods for transmitting redirecting information are available, this expression specifies that they must be taken from the Diversion Header when providing them to the call control.

Mode: Interface SIP

Step	Command	Purpose
1	[name] (if-sip)[if-name]#address-translation incoming-call calling-redir diversion-header	Enables or disables extracting of the redirection information from the Diversion Header.

SIP Calling Party Number Handling

On a SIP interface, up to three calling party numbers can be sent. These three numbers can be set at the following SIP headers:

- **From** header
- **P-Asserted-Identity** header
- **P-Preferred-Identity** header

For the different calling party number handling on ISDN, see Chapter 63, “ISDN Interface Configuration” on page 806.

SIP REFER Transmission (& ISDN Explicit Call Transfer support)

Additional call transfer support, a push-back mechanism, is enabled by default for SIP interfaces by sending REFER messages.

Trinity detects calls that are looped internally, i.e. calls that leave the device over the same SIP interface over which they enter the device. If an internal loop is detected for a SIP interface, Trinity sends a REFER message to push back the call to the connected network as soon as the call is connected.

ISDN interfaces react similarly to internally looped calls. The push-back mechanism is enabled by default for ISDN interfaces (BRI ports) by accepting or rejecting explicit call-transfer (ECT) invocations. An ISDN phone that is connected to a BRI port and that has two active calls can send an ECT invocation to connect the two calls inside the device. An ISDN interface can be configured to accept or reject ECT invocations.

Trinity detects calls that are looped internally, i.e. calls that leave the device over the same ISDN interface over which they enter the device. If an internal loop is detected for an ISDN interface bound by an ISDN user port, Trinity sends an explicit call-transfer (ECT) to push back the call to the connected network as soon as the call is connected. An ISDN interface can be configured to emit ECT invocations.

Figure 119 on page 813 shows an example scenario where a SIP network connects two devices to give a home office access to a PBX in the central office.

The phone in the home office has two active calls to other subscribers of the PBX in the central office. The user wants to connect the other two participants and (a) sends an explicit call-transfer invocation to the device HO. The device HO internally connects the two calls and sends a DISCONNECT message to the phone for both calls. In a second step (b) the firmware on HO detects an internal loop. Both call legs are connected to the same network. In this example, both call legs are handled by the same SIP gateway. The firmware on device HO sends a REFER message to device CO, which connects the two call legs internally and sends a BYE message to

the device HO. (c) Again the firmware of CO detect an internal loop. This time the call legs are handled by the same SIP interface, connected to the PBX. Since the ISDN port is a user port it sends an explicit call-transfer invocation to the PBX (d), which connects the call and sends the device CO a DISCONNECT message for both calls. During all these push back operations the data path of the two participants keeps connected.

The push back mechanism over ISDN (using ECT) and SIP (using REFER) works independently of the protocol that invoked the call-transfer.

The push-back mechanism can be configured on each interface separately. Per default push-back is enabled for ISDN and SIP interfaces. You only have to change the configuration if you don't want internally looped calls to be pushed back to the network. The configuration command **[no] call-transfer accept** configures if an incoming call-transfer request (e.g. ECT or REFER) shall be accepted. The configuration command **[no] call-transfer emit** configures if a call reaching the device over this interface and leaving the device over this interface shall be pushed back to the network, i.e. if a call-transfer request (ECT or REFER) shall be sent.

The following procedure disables the push-back mechanism on the ISDN interface connected to the PBX. No ECT invocation is sent when a call is detected that is looped internally.

Mode: Interface ISDN

Step	Command	Purpose
1	[node](if-isdn)[if-name]#no call-transfer emit	Disables the ISDN push back mechanism.

The following procedure disables the push-back mechanism on a SIP interface. No REFER message is sent when a call is detected that is looped internally.

Mode: Interface SIP

Step	Command	Purpose
1	device(if-sip)[if-name]#no call-transfer emit	Disables the SIP push back mechanism.
2	device(if-sip)[SWITCH.IF_SIP0]#call-transfer expires <seconds>	Defines the expires timeout for a call transfer using SIP refer. If a call transfer will last longer than the expires time, the call(s) will be closed by the Patton device.

AOC Over SIP (Optional)

This enhancement sends AOC (Advice of Charge) information transparently from ISDN to SIP and vice versa. AOC-D elements are hex-encoded and sent as application/QSIG content in SIP INFO messages during a session.

Mode: Interface SIP

Step	Command	Purpose
1	[name] (if-sip)[interface]#[no] aoc-d accept	Enables or disables the reception of SIP Info messages containing AOC-D elements and propagate charging information to adjacent peer.
2	[name] (if-sip)[interface]#[no] aoc-d emit	Enables or disables the sending of SIP Info messages with AOC-D elements containing charging information from adjacent peer. If no charging information is available, no message is sent.

The following commands have to be used to receive and send Advice Of Charge in ASN1 or XML format. Please note that ASN1 format is only supported during a call.

Mode: Interface SIP

Step	Command	Purpose
1	[node](if-sip)[IF_SIP]#[no] aoc-s { accept emit }	Enable/disable the SIP interface AOC /emit feature at the beginning of a call. Default: disabled
2	[node](if-sip)[IF_SIP]#[no] aoc-d { accept emit }	Enable/disable the SIP interface AOC accpt/emit feature during a call. Default: disabled
3	[node](if-sip)[IF_SIP]#[no] aoc-e { accept emit }	Enable/disable the SIP interface AOC accpt/emit feature at the end of a call. Default: disabled
4	[node](if-sip)[IF_SIP]#[no] aoc-format { asn1 xml }	Sets the SIP interface AOC format. Default: asn1

Enabling the Session Timer (Optional)

The gateway implements the SIP session timer feature, which is currently only defined in SIP draft standards. The session timer feature allows a gateway to check periodically during a call, if the remote gateway is still alive and if the call is still connected on the remote gateway. You can enable this feature using the command shown below. If the session timer feature detects that the call is no longer connected on the remote side, it will also drop the call locally.

Mode: Interface SIP

Step	Command	Purpose
1	[node](if-sip)[if-name]#[no] session-timer <seconds>	Enables/Disables the session timer with a refresh period of the specified number of seconds. Default: disabled

Enabling the SIP Penalty-box Feature (Optional)

Normally a SIP call is started by sending an INVITE request to the destination. If that destination is unavailable for any kind of reason, it could take some time to detect that this destination is failing. During that time the party who initiated the call is waiting for the ring-back tone as feedback. To try an alternative path to reach the called destination it is practically too late because the caller gave up during that time. It is now possible to determine the reachability of destinations ahead of actual calls. For a failing destination an alternative routing path to the called end user could be issued immediately without having a timeout. This is done by periodically sending SIP OPTION requests to the configured remote address. If we receive an answer for that OPTIONS request the destination is still alive and calls are forwarded toward it. For the reachability the content of the answer does not matter, neither if the answer is a failing or successful one. If there is no answer during the configured timeout, the remote destination is added to the penalty-box. For a certain time such destinations are considered not reachable, no requests are sent to them and calls can be redirected immediately without timeout. This could happen for example with a hunt-group service in the call-control.

Mode: Interface SIP

Step	Command	Purpose
1	[node](if-sip)[if-name]#[no] penalty-box	Enables/Disables the SIP penalty box feature. Default: disabled

Now two new Action Script events are also generated with the penalty box feature:

- `OPTIONS_TARGET_REACHABLE`
- `OPTIONS_TARGET_NOT_REACHABLE`

`OPTIONS_TARGET_REACHABLE` will be triggered when the Patton device receives an answer to the SIP OPTIONS sent to the remote for the first time. The `OPTIONS_TARGET_NOT_REACHABLE` event is triggered when the first event has already been triggered and no answer is received by the SIP OPTIONS.

These two new Action Script events can be used to change the configuration dynamically according to the reachability of SIP remotes.

1. Configure the penalty-box

Mode: context cs <cs-name>

Step	Command	Purpose
1	[node](ctx-cs)[<cs-name>]# interface sip <ifName>	Creates/Enters a SIP interface.

Step	Command	Purpose
2	[node](if-sip)[<cs-name>.<if-name>]# penalty-box sip-option-trigger interval <interval> timeout <timeout>	Enables the penalty-box with SIP OPTIONS.

2. Configure the Action Script (see section “SIP Interface Events” on page 81).

Initiating a New SIP Session for Redirected SIP Calls (Optional)

Normally, if a SIP call is redirected to a different location by receiving a SIP 3xx response, only the request header is replaced in the original INVITE message and the INVITE is sent to the new destination. Using the following procedure, the SIP gateway can be forced to create a completely new session for forwarded calls.

Mode: Interface SIP

Step	Command	Purpose
1	[node](if-sip)[if-name]#[no] new-session-after-redirect	Enables/disables creation of a new session after a redirect response. Default: disabled

Rerouting Calls from SIP (Optional)

The **call-reroute** command allows to redirect calls to other mediums like FXS or ISDN through the call-control.

Mode: Interface SIP

Step	Command	Purpose
1	[node](if-sip)[if-name]#[no] call-reroute accept	Enables or disables the rerouting to other networks.

Configuring the SIP Hold Method (Optional)

There are different ways to set a remote SIP subscriber On Hold. This command specifies which method the device uses to indicate this call state. In receive direction, all of them will be accepted.

Mode: Interface SIP

Step	Command	Purpose
1	[node](if-sip)[if-name]#hold-method {direction-attribute zero-ip}	Configures the method to be used to signal the hold state. Default: zero-ip
2	[node](if-sip)[if-name]#hold-method direction-attribute { sendonly inactive }	Configures the <i>direction-attribute</i> hold method. Default: sendonly

Configuring SIP Overlap Dialing (Optional)

When the sending feature of overlap-dialing is enabled, the user part of the to-header and request URI is taken from the called party number. Trinity sends for each update of the called party number a new INVITE message

with an updated to-header and request URI, but uses the same call-id and from-tag as the original invite. As soon as one of the INVITE requests gets a response establishing an early-dialog the overlap dialing is finished. All other INVITE requests which are still pending are canceled at this time and additional digits are transported through the established session. If all INVITE requests get a negative answer and the sip interface receives no updates of the called party number for the digit-collection timeout time, the overlap dialing fails and the call is dropped.

When overlap dialing acceptance is enabled, an incoming INVITE request is answered with a “100 Trying” message and the call is forwarded to the called destination. When an INVITE is received with the same call-id and from-tag as an already pending INVITE request, the pending request is always released with a “484 address incomplete” message. Then, the new request is answered with a “100 Trying” message and the additional digits received with the new INVITE are forwarded to the called destination. When the called destination signals to establish an early-dialog the overlap acceptance period ends and the last received INVITE request is answered with the according answer. When the called destination drops the call, the overlap dialing session ends with a final negative response. An additional INVITE, coming in after overlap dialing has ended, is treated as an independent request and creates a new call.

Mode: Interface SIP

Step	Command	Purpose
1	[node](if-sip)[name]# [no] overlap-dialing new-transaction accept	Configures the sip interface to accept overlap dialing.

Mode: Interface SIP

Step	Command	Purpose
1	[node](if-sip)[name]# [no] overlap-dialing new-transaction emit	Configures the sip interface to emit overlap dialing.

Configuring PRACK for Reliable Provisional Responses (optional)

Reliable Provisional Responses are supported according to RFC3262. Reliability is achieved by issuing a PRACK message upon reception of a provisional response.

Note the following conditions:

- It is the UAC that issues an INVITE message, which will also issue a PRACK message upon reception of a reliable provisional response.
- It is the UAS that receives an INVITE request, which will also receive a PRACK request upon issuing a reliable provisional response.

Two configuration commands in the SIP interface of 'context cs' allow specifying the behavior on both client and server sides. The UAC indicates its behavior with a 100rel tag in the Supported or Required header field of the INVITE message.

UAC Side:

Step	Command	Purpose
1	<code>[node](if-sip)[name]# [no] prack emit {supported required}</code>	Enables PRACK for outgoing calls when acting as UAC Default: disabled

- **supported:** The peer may send provisional responses reliably.
- **required:** The peer must send provisional responses reliably. The call fails if the peer is not willing or fails to do so.

UAS Side:

Step	Command	Purpose
1	<code>[node](if-sip)[name]# [no] prack accept {supported preferred required}</code>	Enables PRACK for incoming calls when acting as UAS Default: disabled

- **supported:** Provisional responses are sent reliably if it is required by the peer.
- **preferred:** Provisional responses are sent reliably if it is supported or required by the peer.
- **required:** Provisional responses are sent reliably. The peer must support reliable provisional responses. The call fails if the peer does not support reliability for provisional responses.

The following table summarizes the UAS' behavior:

		Remote Peer		
		no reliability	supported	required
Local configuration	no prack accept	non-reliable	non-reliable	failure
	prack accept supported	non-reliable	non-reliable	reliable
	prack accept preferred	non-reliable	reliable	reliable
	prack accept required	failure	reliable	reliable

Limitations:

The following scenarios are not supported:

- Reliable Provisional Responses and PRACK are not supported for any re-INVITE scenarios.
- If an INVITE request is sent to a stateless proxy who forks the request to several UAS, then the SIP protocol on the device cannot correctly handle all Reliable Provisional Responses it receives from all the UAS. Hence, such a scenario will not work.

Configuring History-Info in SIP

For SIP to SIP calls the History-Info header is now supported according to RFC 7044. This includes reception and forwarding, the generation of new headers for changed request uri and generation of new headers for redi-

rected calls. The History-Info header is only treated in INVITE messages and its responses. The support of the History-Info header is announced with the "histinfo" tag in the Supported header of the initial INVITE.

Mode: context cs / interface sip

Step	Command	Purpose
1	node(if-sip)[<name>]#[no] history-info emit	Enables or disables the sending of History-Info headers in INVITE messages and responses to INVITE messages. Default is disabled.

Enabling NAT Traversal for SIP INVITE Messages

The following command enables Nortel-type NAT traversal for SIP INVITE messages. This command sends SIP PING requests every 55 seconds after a successful registration to keep NAT open.

Mode: Interface SIP

Step	Command	Purpose
1	[node](if-sip)[<name>]#[no] nat-traversal nortel	Enables Nortel-type NAT traversal for SIP INVITE messages.

Accepting Re-invite After Reboot

To distinguish the re-invite from an initial invite, Trinity examines if a header-parameter 'tag' is present in the To header. If the To-tag is present and the re-invite does not match any ongoing call, it is rejected with a '481 Call/Transaction Does Not Exist' answer. This behavior can be changed to accept an initial invite with a To-tag, which goes against RFC3261 and should therefore be avoided.

Mode: SIP

Step	Command	Purpose
1	[node](if-sip)#[no] check-to-tag	Specifies if initial invites with To-tag should be rejected. Default: <i>enabled</i>

Chapter 53 **SIP Survivability**

Chapter contents

Introduction	666
Background	666
Operation Principle	666
SIP-Survivability Configuration Tasks.....	668
Set up an Ethernet bridge.....	669
Create a Location Service.....	669
Create a SIP Gateway	670
Configure Call-Routing.....	670
Configure the SIP-Survivability Service	671
Configure the SIP-Survivability Agent.....	671
Essential Configuration	672
Optional commands for tuning the observation of SIP messages	672
Optional commands for tuning the observation of DNS messages	674
Inspect the SIP-Survivability Status	677
Debugging the SIP Survivability Agent	680

Introduction

This chapter describes how to set up the SIP survivability agent that is built into the Patton device. The survivability agent passively monitors ongoing SIP transactions between a local (enterprise) network and a remote (hosted) SIP service on an Ethernet bridge and learns the contact addresses of registering SIP phones and the registered-to SIP services. When a remote SIP service becomes unreachable, the survivability agent takes over the responsibility of the service and connects the SIP sessions locally, allowing station-to-station, emergency, and external breakout calls.

Background

A recent trend for enterprises is to out-source their SIP-based Private Branch Exchange (SIP-PBX) into the cloud. VoIP providers have started to offer hosted PBX and UC services where internal and external phone calls are switched in the provider's datacenter. However, the advantage of delegating the operation of an enterprise PBX to a professional service provider comes with a down-side: A hosted service running in a remote datacenter is naturally susceptible to outages of the enterprise's WAN link to the Internet. This leads to the following negative consequences: First, station-to-station calls within the company are no longer possible during a WAN-link outage. Second, and more importantly, emergency E911 calls are not delivered any longer if the WAN-link is down or if the hosted PBX service is unreachable for another reason.

Several survivability solutions have yet been proposed to mitigate this problem: One approach is to set up a backup WAN link that becomes active if the primary link goes down. This approach provides a redundant path on the network layer (IP). However, such a layer-three redundancy solution is not able to prioritize emergency VoIP calls over data traffic, for example. Since the redundant link often exhibits a lower bandwidth than the primary link it is of utter importance to prioritize important traffic such as emergency calls.

Another, typical solution suggests installing a Session-Border Controller (SBC) at the edge of the enterprise network. The SBC either acts as an application-level gateway (e.g. SIP back-to-back User Agent (B2B-UA)) or as a SIP application proxy. By recognizing SIP signaling flows, an SBC is able to detect that a remote SIP service is no longer reachable and switch the SIP sessions over an alternate route, e.g. over a backup WAN link or via a telephony gateway to the PSTN network. This is a reliable way of granting that emergency calls are still possible. The problem with this approach is that all participating endpoints within the enterprise have to route the SIP sessions over that SBC. That is, SIP phones must be re-configured to connect to the SBC instead of the hosted SIP service or use the SBC as an outbound proxy. Experience showed that many enterprises first deploy a hosted telephony service and take care of redundancy only later, such that potentially thousands of phones must be reconfigured.

What is required by many enterprises is a network function that can be looped in between the enterprise LAN and the Internet access router and observes the SIP traffic between the local SIP endpoints (e.g. phones) and the remote SIP servers in a non-intrusive way, i.e. by not modifying any of the passing SIP messages. The device should automatically learn the addresses of the SIP services as well as the contact addresses of the local endpoints in the LAN. It should continuously probe whether the detected SIP services are reachable or not. Only in the event of an unreachable SIP service should the device interact with the SIP endpoints in order to connect local station-to-station calls (from LAN endpoint to LAN endpoint) and provide an alternative route for emergency and other external calls (e.g. over a PSTN breakout gateway).

Operation Principle

The SIP survivability agent is a software component built into the Trinity software. As depicted in [figure 106](#) on page 667, it is connected to an Ethernet bridge that is looped-in between the enterprise LAN and the Internet Access Device (IAD) or enterprise edge router. Looped in between the enterprise LAN and the IAD, the

agent observes SIP traffic, detects if a SIP service becomes unreachable, and in this case, takes over all SIP calls, connects them locally or places outbound calls over an alternative network (e.g. PSTN).

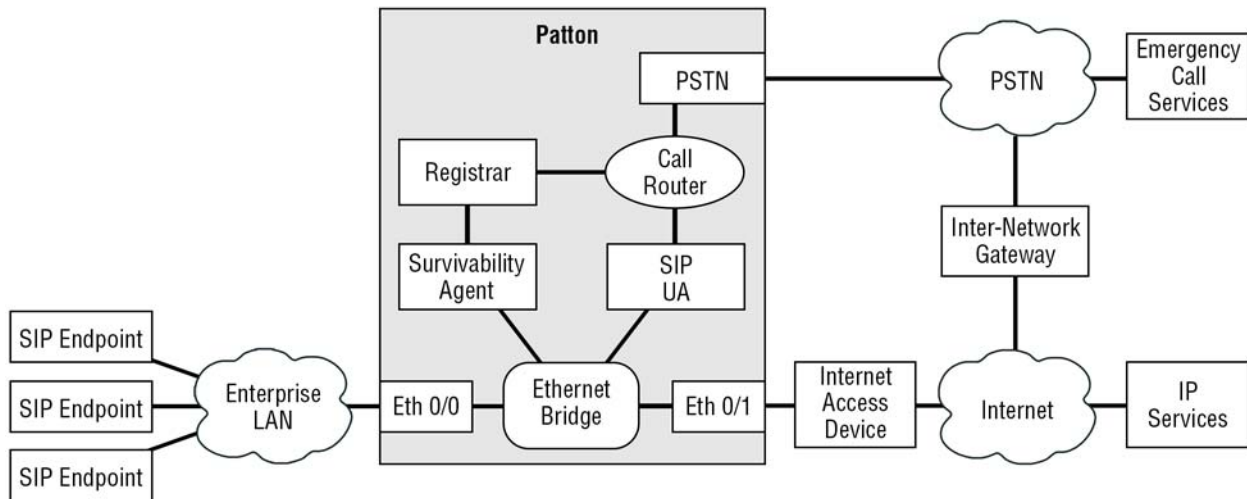


Figure 106. The Patton device contains a survivability agent software component.

The Patton device containing the Survivability Agent supports the following features:

- The Patton device transparently passes all traffic, including all SIP packets, from the enterprise LAN to the Internet and vice versa through an Ethernet bridge. In normal mode (if the SIP service is reachable), the Patton device does not modify any of the bridged packets to achieve full service transparency.
- The Patton device sends a duplicate of all forwarded SIP packets internally to the Survivability Agent module.
- The Survivability Agent learns from the observed REGISTER requests
 - The set of all SIP services¹ in the Internet from the Request-URI, and
 - The set of all SIP clients in the enterprise LAN from the Contact-header.
- The Survivability Agent continuously checks the reachability of the learned SIP services by sending OPTIONS pings and updates a local database. Reachability states: unknown, up, down.
- The Survivability Agent maintains a registration database of all learned SIP clients that acts as backup registrar.

1. Note that we are making a distinction between a "SIP service" and a "SIP server". We use the term "SIP server" for a physical or virtual machine running the SIP protocol, acting as call switch or PBX. A SIP server typically is addressed by a unique IP address or hostname (e.g. sip1.example.com). A "SIP service" on the other hand is offered at a Fully-Qualified Domain Name (FQDN), typically resolved with a DNS service record (e.g. example.com). Requests to a SIP service may be distributed to multiple servers for redundancy and load-balancing reasons. The SIP Survivability feature checks the reachability of the "SIP service", i.e. checks whether OPTIONS requests sent to the service URI are answered by the service. Hence, even if a SIP server is down, the service may still be available if there is a redundant server.

- If a SIP request (e.g. INVITE) is observed for a SIP service that is reachable, the Survivability Agent does not interfere at all.
- Otherwise, if a SIP request is observed for a SIP service that is unreachable, the Survivability Agent responds appropriately:
 - REGISTER requests are answered with "200 OK" (no authentication required).
 - CANCEL requests are answered with "200 OK".
 - ACK requests are ignored.
 - INVITE requests are answered with "302 Temporarily Moved", redirecting the call to the local SIP UA. This redirected call ends up in the Trinity Call-Router (see chapter 49, “[Call Router Configuration](#)” on page 561, which can be configured to lookup the backup registration database to connect local station-to-station calls or to send a call to an alternative network such as POTS, ISDN, another SIP trunk, etc.
- The Survivability Agent also sends a duplicate of all forwarded DNS responses to the Survivability Agent module.
- The Survivability Agent learns from the observed DNS responses
 - The set of all name servers in the Internet
 - The resource records resolved by any of the clients in the LAN
- The Survivability Agent matches the learned DNS resource records with the observed SIP services to reduce the set of cached DNS records and archives them to persistent memory. This allows the Patton device to act as a DNS caching server in case of a WAN link outage even if after a reboot the WAN link is still down.

Note In contrast to other SIP survivability solutions, our solution neither is a B2B-UA nor a SIP proxy. That is, if a SIP service is reachable, the Patton devices does not modify passing SIP packets in any way. The advantage of this approach is that the Patton device does not have to understand any SIP extension (e.g. proprietary X-headers). Any proprietary SIP header or non-standard message-exchange behavior is supported. Only in the event of an unreachable SIP service does the Trinity device interfere with the SIP endpoints, and provides the service of connecting local calls and placing emergency and other outbound calls over an alternative telephony network.

SIP-Survivability Configuration Tasks

Several software components of the Patton device must work together to offer the desired survivability and fallback-gateway functionality. The following tasks must be executed to configure all involved components:

1. Set up an Ethernet bridge (see [page 669](#))
2. Create a Location Service (see [page 669](#))
3. Create a SIP Gateway (see [page 670](#))
4. Configure call routing (see [page 670](#))
5. Configure the SIP-Survivability service (see [page 671](#))

All but the last task are configuring non-survivability components that are documented in other chapters of this document. However, we provide a complete example here to help for a quick deployment.

Set up an Ethernet bridge

One feature of the SIP Survivability Agent is to transparently pass traffic from the enterprise LAN to the Internet. This is done by installing a software-based Ethernet bridge between the LAN and the IAD (see [figure 106](#) on page 667). For example, if the IAD is connected to Ethernet port 0/0 and the enterprise LAN is connected to Ethernet port 0/1 (or vice-versa), the following configuration will set up layer 2 bridging between those ports:

```
context ip ROUTER

    interface LAN
        ipaddress LAN dhcp

context bridge

    bridge-group BRIDGE
        no shutdown
        bind interface ROUTER LAN

port ethernet 0 0
    bind bridge-group BRIDGE
    no shutdown

port ethernet 0 1
    bind bridge-group BRIDGE
    no shutdown
```

Note This example uses DHCP to allocate an IP address from the LAN DHCP server. This should also provide the Patton device with a default route and a list of name-servers to use. If DHCP is not enabled in your LAN you should configure a static IP address; but don't forget to also configure a default route and a list of reachable name-servers manually.

Create a Location Service

The location service acts as the backup registrar, storing the contact addresses of all SIP phones in the LAN. This backup registration database is filled automatically by the Survivability Agent while listening for the traffic traversing the Ethernet bridge.

```
location-service LS_SURV
    match-any-domain

    identity-group DEFAULT

    registration inbound
```

We are not restricting the domain name ('match-any-domain') such that registrations to all SIP servers can be cached locally. Like this the Survivability Agent is able to track registrations for multiple SIP domains at the same time. If you want to restrict the registered phones, please configure the domain-name accordingly.

Create a SIP Gateway

The SIP Gateway acts as a fallback gateway. It will only receive calls that are actually sent to the cloud-based PBX if that SIP service is unreachable. If the SIP service is reachable, this gateway will not receive any calls as they are transparently passed to the Internet via the Ethernet bridge.

```
context sip-gateway SIP_GW_SURV
  bind location-service LS_SURV      # created in step 2

  interface IF
    transport-protocol udp+tcp 5060
    no transport-protocol tls
    bind ipaddress ROUTER LAN LAN    # created in step 1

context sip-gateway SIP_GW_SURV
  no shutdown
```

Configure Call-Routing

If a call reaches the SIP Gateway configured above, we are in fallback mode. With the call-router you can specify how to handle such a call. You have to create one SIP interface (e.g. IF_SIP_SURV) that is bound to the SIP Gateway (SIP_GW_SURV from step 3) over which fallback calls are received. Typically, we want to route those calls to a hunt-group that first checks in the local backup registrar (LS_SURV from step 2) for the destination address of the call. If a phone has been registered to the call's destination number it will be routed back to the incoming interface. Otherwise, we may treat the call as external call and route it to a breakout interface (IF_ISDN_BREAKOUT), for example.

```
context cs SWITCH
  no shutdown

  interface sip IF_SIP_SURV
    bind context sip-gateway SIP_GW_SURV  # created in step 3
  route call dest-service SVC_HUNT_LOCAL_THEN_OUT

  interface isdn IF_ISDN_BREAKOUT
    route call dest-service SVC_LOOKUP

  service sip-location-service SVC_LOOKUP
    bind location-service LS_SURV        # created in step 2

  service hunt-group SVC_HUNT_LOCAL_THEN_OUT
    route call 1 dest-service SVC_LOOKUP
    route call 2 dest-interface IF_ISDN_BREAKOUT

port bri 0 0

q921
  protocol pmp
  uni-side auto

q931
  uni-side user
bind interface SWITCH IF_ISDN_BREAKOUT
```

```
port bri 0 0
shutdown
```

Configure the SIP-Survivability Service

Finally, we have to enable the SIP Survivability service and tell it over which SIP gateway we want to receive fallback calls.

```
sip-survivability
  bind context sip-gateway SIP_GW_SURV    # created in step 3
  no shutdown
```

These two are the only essential commands you have to configure in the 'sip-survivability' mode. However, there are more commands in this mode that can be used to tune the Survivability Agent's functionality. Those commands are discussed in detail in the next section.

Configure the SIP-Survivability Agent

The configuration commands for the SIP-Survivability Agent are split in three categories:

- Essential commands,
- Optional commands for tuning the observation of SIP messages, and
- Optional commands for tuning the observation of DNS messages.

Essential commands are those that you have to configure in order for the survivability service to work. You (a) have to bind the SIP-Survivability Agent to a valid SIP Location Service and (b) enable the feature by executing the 'no shutdown' command as shown in step 5 above.

By enabling the service, the SIP Survivability Agent starts listening to SIP messages passing the configured Ethernet bridge (see step 1 above). The remaining commands are used to tune how this observation shall work. Note that next to SIP messages the Patton devices also observes DNS messages since it has to be able to answer DNS queries for local SIP phones in case the WAN link is down.

Essential Configuration

This procedure describes how to Bind to an existing SIP Gateway and enable the survivability service.

Mode: Configure

Command	Purpose
node(cfg)# sip-survivability	Enters the survivability configuration mode. All commands related to the Survivability Agent are located in this mode.
node(sip-surv) # [reset no] bind context sip-gateway gw-name	<p>Binds to an existing SIP gateway called gw-name. The gateway must fulfill the following requirements:</p> <ul style="list-style-type: none"> • The bound SIP gateway must be bound to a Location Service • There must exist one gateway interface that is bound to an IP interface address. • An Ethernet bridge-group must bind to that IP interface. • There must exist at least one SIP interface in context cs that is bound to the SIP gateway. • The bound SIP gateway must be enabled ('no shutdown')
node(sip-surv) # [reset no] shutdown	<p>Starts or stops the SIP Survivability Agent.</p> <p>Note The internal database of learned SIP server and phone addresses as well as the database of cached DNS records is NOT flushed by shutting down the Survivability Agent. Use the 'erase sip-survivability' command to flush the cache (see below).</p>

Optional commands for tuning the observation of SIP messages

The Survivability Agent sends periodic OPTIONS requests to the learned SIP services to check whether they are reachable. The interval of those pings as well as other timing parameters are configured by the following commands in the 'sip-service' sub-mode.

In most networks, such a parameter tuning is not required for the survivability feature to work. However, the time between a WAN link outage and the calls being rerouted through the SIP gateway may be optimized by reducing the interval and the request timeout of the OPTIONS pings. Choosing the right parameters is a tradeoff between reaction time and additional bandwidth and load for the SIP services by sending frequent OPTIONS pings.

Procedure: Tune SIP observation parameters

Mode: SIP Survivability

Command	Purpose
node(sip-surv) # sip-service	Enters the configuration mode to configure SIP observation parameters.
node(ss-sip-service) # [reset] ports udp (<i>port-range</i> all)	<p>Configures a range of UDP ports on which the observed SIP services are listening for incoming SIP requests. Most SIP services listen on UDP port 5060, which is why this is the default configuration.</p> <ul style="list-style-type: none"> • <i>port-range</i>: Either a single port number (default: 5060) or a range of ports (e.g. 5060..5069) • all: Listens on all UDP ports (1..65535) for SIP messages. <p>Warning: This leads to a severe performance degradation of the Ethernet bridge and of the device in general. You may want to use this setting initially to let the device learn all SIP services on all UDP ports but you should restrict the port range once you know the ports in use. (See the 'show sip-survivability' command below to print a list of all learned SIP services.)</p>
node(ss-sip-service) # [reset no] options-ping [interval <i>interval</i>] [request-timeout <i>timeout</i>]	<p>When this command is enabled (default), the Survivability Agent periodically sends OPTIONS requests to all learned SIP services to check whether they are reachable. This command configures the interval of those pings and the time between sending an OPTIONS request and expecting a response.</p> <ul style="list-style-type: none"> • <i>interval</i>: Interval in seconds of sending OPTIONS ping requests to all learned SIP services (default: 30s) • <i>timeout</i>: If after sending an OPTIONS request to a SIP service the Patton device does not receive a response within this timeout in seconds, the failure counter is incremented (default: 10s). If the failure counter reaches the value configured with the 'failure-count' command (see below) the SIP service is considered unreachable and all INVITE messages to that service will be redirected to the bound SIP gateway. <p>Note Some SIP services don't like frequent OPTIONS requests. Please ask your cloud PBX provider for a tolerable interval.</p>

Command	Purpose
node(ss-sip-service) # [reset] failure-count count	<p>Configures after how many unanswered OPTIONS requests a SIP service is considered unreachable. The default setting is 1, which means that the first OPTIONS ping that remains unanswered will cause all INVITE messages to that service to be redirected to the bound SIP gateway.</p> <p>Note The OPTIONS request itself is re-transmitted according to the SIP standard (e.g. after 500ms, 1s, etc.) That is, a single loss of an OPTIONS packet usually does not lead to the Survivability Agent to take over calls. However, if the SIP service suffers from a performance degradation, it may well be that OPTIONS requests are not being answered within the 10 seconds timeout. Please ask your cloud PBX provider for the expected worst-case delays.</p>

Optional commands for tuning the observation of DNS messages

Most SIP services are addressed with a Fully-Qualified Domain Name (FQDN), such as sip:user@example.com, that require resolving the SIP service name (example.com) via the Domain Name Service (DNS). If an enterprise does not run its own caching name server within the LAN SIP phones cannot resolve the SIP service name to a SIP server IP address in case of a WAN link failure and hence, the phones will not send INVITE messages over the Patton device.

Therefore, the Survivability Agent is equipped with an automatic DNS cache: It passively listens to the DNS traffic passing the Ethernet bridge and populates its cache with the resource records found in the observed DNS responses. It optionally pings each DNS server learned from that traffic with queries. If a remote DNS server does not respond to a query ping or to an observed query that passes the Ethernet bridge, it considers that DNS server unreachable and responds to queries from the LAN from its local cache. The cache can be made persistent such that it survives a reboot. This is helpful if the enterprise suffers from a power outage where after the Patton device boots up again, it shall provide local SIP connectivity and breakout as long as the WAN link is down.

The following commands in the 'name-server' sub-mode within the 'sip-survivability' mode may be used to tune timing parameters of the DNS cache. In most networks, the default settings should be fine though.

Procedure: Tune DNS-cache parameters

Mode: SIP Survivability

Command	Purpose
<code>node(sip-surv) # name-server</code>	Enters the configuration mode to configure DNS-cache parameters.
<code>node(ss-name-server) # [reset] ports udp (port-range all)</code>	<p>Configures a range of UDP ports on which the observed name servers are listening for incoming DNS queries. Most DNS servers listen on UDP port 53, which is why this is the default configuration.</p> <ul style="list-style-type: none"> • <i>port-range</i>: Either a single port number (e.g. 53) or a range of ports (e.g. 53..55) • all: Listens on all UDP ports (1..65535) for DNS messages. <p>Warning: This leads to a severe performance degradation of the Ethernet bridge and of the device in general. You may want to use this setting initially to let the device learn all name servers on all UDP ports but you want to restrict the port range once you know the ports. (See the 'show sip-survivability dns' command below to print a list of all learned name servers.)</p>
<code>node(ss-sip-service) # [reset no] passive [request-timeout timeout]</code>	<p>Configures the <i>timeout</i> between a bridged DNS query from the LAN and the observed DNS response from the name server. If the name server does not respond within the specified timeout, the failure counter is incremented. If the failure counter reaches the value configured with the 'failure-count' command (see below) the DNS server is considered unreachable and all DNS queries from the LAN are answered with entries from the local DNS cache.</p> <p>To disable the DNS cache, use the inverted form of the command ('no passive').</p> <p>Note The query-ping option (see below) requires the passive option to be switched on. Hence, if you enable the query-ping command, it automatically options the passive command.</p>

Command	Purpose
node(ss-name-server) #[reset no] query-ping [interval interval] [request-timeout timeout] [forced]	<p>When this command is enabled (default), the Survivability Agent periodically sends DNS dummy queries to all learned name servers to check whether they are reachable. This command configures the interval of those pings and the time between sending an DNS query and expecting a response.</p> <ul style="list-style-type: none"> • interval: Interval in seconds of sending DNS ping queries to all learned SIP services (default: 30s) • timeout: If after sending a DNS query to a name server the Patton device does not receive a response within this timeout in seconds, the failure counter is incremented (default: 10s). If the failure counter reaches the value configured with the 'failure-count' command (see below) the name server is considered unreachable and all DNS queries from the LAN to that name server will be answered from the local DNS cache. • forced: If this option is specified, DNS ping queries are always sent after the specified interval. If this option is not specified, the pings are only sent if no device in the LAN was sending a query to the same name server within the configured interval (default: disabled)
node(ss-name-server) # [reset] failure-count <i>count</i>	<p>Configures after how many unanswered DNS queries a name server is considered unreachable. The default setting is 1, which means that the first query ping that remains unanswered will cause all future queries to that name server to be answered from the local DNS cache.</p>
node(ss-name-server) # [reset no] persistent-cache [sip-related all]	<p>Configures which resource records are stored persistently to survive a device reboot.</p> <ul style="list-style-type: none"> • disabled: If the inverted form of the command is entered ('no persistent-cache') the DNS cache is not stored to persistent memory. After a reboot, the Patton device will not be able to respond to DNS queries from the LAN. • all: All resource records are stored to persistent memory. This setting is not recommended as it could fill the persistent memory of the device quickly. • sip-related: Only stores resource records that are related to SIP services to persistent memory (default).

Inspect the SIP-Survivability Status

The following commands can be executed to inspect which SIP services are being observed and how many SIP clients have been registered and will be connected locally in case of a WAN link failure.

Procedure: Display SIP-Survivability status

Mode: Operator Exec

Command	Purpose
> show sip-survivability [<i>detail detail-level</i>] [<i>continuously</i>]	<p>Displays the status of the SIP-Survivability Agent, which includes a list of observed SIP services and a list of registered SIP endpoints.</p> <ul style="list-style-type: none"> • <i>detail-level</i>: Determines how much information details are printed (default: 1) <ol style="list-style-type: none"> 1: Tabular output of SIP services and registered SIP endpoints 2: More information about the configuration of the feature and status of the SIP services 3: Full status information • full-detail: Short hand for 'detail 3' • continuously: Reprints the output every second until Ctrl-C is pressed

Example: Show License Server status and number of connections

```
> show sip-survivability
SIP Survivability
=====

Administrative status:      enabled
Operational status:       running

Observed SIP Servers
=====

Server          Status  Duration   IP Address
-----
sip:sip.inalp.com up      16h 34m 34s 217.11.220.250

Local Registrations
=====

User   Contact Address
-----
53712  sip:53712@192.168.168.101:5060
53711  sip:82371954@192.168.168.102:63499
```

Example: Use the show command to see whether the required license is installed.

```
> show sip-survivability
SIP Survivability
=====

Administrative status:      enabled
Operational status:       stopped
Reason:                    SIP Survivability license is absent
```

Procedure: Display DNS-Survivability status

Mode: Operator Exec

Command	Purpose
> show sip-survivability dns [detail detail-level] [continuously]	<p>Displays the status of the DNS-Survivability Agent, which includes a list of observed name servers.</p> <ul style="list-style-type: none"> <i>detail-level</i>: Determines how much information details are printed (default: 1) <ol style="list-style-type: none"> 1: Tabular output of name servers and whether they are reachable 2: More information about the configuration of the feature and status of the name servers 3: Full status information full-detail: Short hand for 'detail 3' continuously: Reprints the output every second until Ctrl-C is pressed

Example: Show License Server status and number of connections.

```
> show sip-survivability dns
DNS Survivability
=====

Administrative status:      enabled
Operational status:       running

Observed DNS Servers
=====

Server          Status  Duration
-----
172.16.30.10:53 up      1m 3s
172.16.30.20:53 up      41s
8.8.8.8:53      up      3s
```

Procedure: Display DNS cache

Mode: Operator Exec

Command	Purpose
> show sip-survivability dns cache [continuously]	<p>Displays the DNS cache and whether the cached resource records are archived to persistent memory or not.</p> <ul style="list-style-type: none"> • continuously: Reprints the output every second until Ctrl-C is pressed

Example: Show DNS cache.

```
> show sip-survivability dns cache
DNS Cache
=====

Server: 172.16.30.10:53
-----

SIP.INALP.COM. NAPTR IN (archived)
SIP.INALP.COM. A IN (archived)

Server: 172.16.30.20:53
-----

SIP.INALP.COM. NAPTR IN (archived)

Server: 8.8.8.8:53
-----

PATTON.IO. A IN
PATTON.IO. AAAA IN
```

Procedure: Flush the SIP-Survivability cache (SIP services, SIP registrations and DNS cache)

Mode: Operator Exec

Command	Purpose
<pre>> erase sip-survivability (sip dns all)</pre>	<p>Flushes the SIP-survivability cache.</p> <ul style="list-style-type: none"> • sip: Clears the list of observed SIP services and registered SIP endpoints. If the SIP-survivability feature is still enabled, the cache will soon be populated again with the learned SIP services. • dns: Clears the list of observed name servers and flushes the local DNS cache. IF the SIP-survivability feature is still enabled, the cache will soon be populated again with the learned name servers and stored resource records. • all: Clears both, the SIP and the DNS lists and caches.

Example: Erase all caches related to the SIP Survivability Agent.

```
>erase sip-survivability all
>
```

Debugging the SIP Survivability Agent

Enable the following debug monitor to print a live trace of the SIP Survivability Agent in action.

Procedure: Enable the SIP-Survivability debug monitor

Mode: Operator Exec

Command	Purpose
<p>> [no] debug sip-survivability (detail <i>detail-level</i> full-detail)</p>	<p>Enables/Disables the print-out of debug messages related to the SIP-Survivability Agent, both SIP- and DNS-related.</p> <ul style="list-style-type: none"> • <i>detail-level</i>: <ol style="list-style-type: none"> 1: New SIP services and new name servers observed, changes of the reachability state of the services and servers, archived DNS cache entries. 2: Additionally, state-machine events and actions 3: Additionally, one-line messages of received and sent messages and how they are processed. <p>Note SIP messages are printed only via the 'sip-transport' monitor.</p> <p>4: More information about how the messages are processed.</p> <p>Note SIP messages are printed only via the 'sip-transport' monitor.</p> <p>5 (full-detail): Full dump of received and transmitted DNS messages.</p> <p>Note SIP messages are printed only via the 'sip-transport' monitor.</p>

Note The sip-survivability debug monitor is also included in the 'essential-voip' debug monitor. If you want to print all debug messages related to any VoIP subsystem, use the 'debug essential-voip' command.

Note Observed SIP messages are printed only with the 'sip-transport' debug monitor, which however is included in the 'essential-voip' debug monitor as well.

Chapter 54 TLS Configuration for SIP

Chapter contents

Introduction	684
About Transport Layer Security (TLS)	684
TLS configuration task list.....	684
Install license	684
Configure URI scheme	684
URI Scheme for Calls	685
SIP	685
SIPS	685
Transparent.....	685
URI Scheme for Registration	686
SIP	686
SIPS	686
URI Scheme for Re-routed Calls	686
Rerouted	686
Original.....	686
URI Scheme for Transferred Calls	687
Configure Time	687
Configure Public-Key Infrastructure (PKI)	687
About TLS Profiles	687
Basic Operation Principle of TLS	688
TLS without authentication (default behavior)	688
TLS authentication with self-signed certificates	688
TLS Authentication with 3rd Party Signed Certificates	689
TLS Profile Default	689
Configure TLS Profile	689
Creating a TLS profile and enter configuration mode	689
Private key	690
Own-certificate chain	690
Trusted Certificates	691
Authentication	692
Cipher Suites	693
Compression	694
Protocol	694
Configure TLS profile usage	694
Configure TLS transport	694
Configure transport enforcement or fallback	695
Configure Non-Default TLS Port Usage	697
Configure SRTP	698
Security consideration: Why you should use mutual TLS authentication	699

IP does not Re-use Connections for Requests in Different Directions	699
Two Scenarios of TLS Authentication Vulnerability	700
Conditions for the Patton device to Re-use Connections	700
Caveat for TLS Profile	701
Showing TLS profile	701
Troubleshooting	702
Check License	702
Check time	702
Check Certificates	703
Check TLS Profile Status	704
Check SIP Gateway Status	706
Check Server Name	706

Introduction

This chapter provides an overview of Trinity's Transport Layer Security (TLS) capabilities for SIP and describes the tasks involved in configuring it.

About Transport Layer Security (TLS)

TLS provides communication security over insecure networks such as the Internet. TLS allows transport connections such as TCP to be safe from eavesdropping and tampering. For this purpose, TLS uses a Public Key Infrastructure (PKI) for authentication and confidentiality of the key exchange. Payload confidentiality is ensured with symmetric encryption using the exchanged keys. Message authentication and integrity is achieved by adding authentication codes to each message.

Using TLS for SIP signaling ensures that the content of the SIP messages is not subject to eavesdropping. This is especially important for SRTP master keys, which are exchanged over SIP/SDP signaling messages. TLS guarantees that the next hop destination is authenticated and that the messages are not altered by a third party.

TLS configuration task list

To configure TLS, perform the tasks in the following sections:

- Install license
- Configure URI scheme
- Configure Time
- Configure Public-Key Infrastructure (PKI)
- About TLS profiles
- Configure TLS profile
- Configure TLS profile usage
- Configure TLS transport
- Configure transport-protocol enforcement or fallback
- Configure non-default TLS port usage
- Configure SRTP
- About security issue: connection re-use
- Troubleshooting

Install license

Voice security is a licensed option for Patton devices. You need to have a "sip-tls-srtp" license installed on your device before you are able to configure and use SRTP and TLS.

Configure URI scheme

The next step for enabling SIP over TLS is to configure the URI scheme for all SIP services that require a secure transmission. The URI scheme for SIP calls is to be configured on the SIP interface in context CS, whereas the URI scheme for SIP registrations is configured on the outbound registration mode in the context

SIP gateway. This separation gives you the flexibility to decide individually whether or not call or registration connections will be encrypted.

URI Scheme for Calls

For calls there are three possible settings for the URI scheme.

- SIP
- SIPS
- transparent

SIP. The SIP URI scheme defines an unsecure SIP URI, i.e., “sip:200@patton.com”. Calls destined to a request-URI with the SIP URI scheme usually are signaled over non-secure transports as UDP and TCP. However, they may be signaled over secure transports such as TLS as well. The transports depend on the availability and preference of transport protocols for reaching the given URI. For example, a DNS lookup for an URI according to the SIP URI scheme may return a list of transport protocols, including TLS. Our protocol preference selection configurable in the call-outbound configuration mode of the location service – then defines which of the transports is preferred.

SIPS. The SIPS URI scheme defines a secure SIP URI, i.e., “sips:200@patton.com”. Calls destined to a request-URI with the SIPS URI scheme must be signaled over the secure transports TLS. (Although there are alternative secure transports protocols, Patton devices only support TLS at the moment).

Transparent. Selects the URI scheme on a per-call basis. The actual URI scheme for an outgoing call is defined by the call-control peer, i.e., over the protocol where the call entered the Patton device. For all protocols other than SIP, when there isn’t any security information – we always use the SIP URI scheme. However, when an incoming SIP call is routed over SIP again, the URI scheme is tunneled transparently through our call-control application. If the requested URI of the incoming call is SIPS and the transport protocol is TLS, then the outgoing call will also be placed using the SIPS URI scheme. However, when the incoming request-URI is SIP or when it arrives over a non-secure transport the resulting URI scheme is SIP.

Mode: Interface SIP

Step	Command	Purpose
1	<code>nodeif-sip)[name]# uri-scheme { sip sips transparent }</code>	Configures the URI scheme for outgoing calls. Default is transparent.

Note If you are using SRTP it is strongly recommended to signal calls over a secure TLS connection by using the SIPS URI scheme. This is because the SRTP session’s master key is exchanged in the SDP portion of SIP signaling messages. If an insecure transport is used for SIP signaling, an eavesdropper may read the SRTP master key and listen to the exchanged RTP voice packets. See “Configure SRTP” below for more information.

URI Scheme for Registration

The URI scheme for outbound registrations can be configured separately from the URI scheme for outbound calls.

Note For registration, the transparent option is not available, because there isn't any incoming source for registrations.

- SIP
- SIPS

SIP. The SIP URI scheme defines an insecure SIP URI, for example “sip:200@patton.com”. The registration messages and incoming calls to the registered contact *may* travel over any secure or non-secure transport.

SIPS. The SIPS URI scheme defines a secure SIP URI, for example “sips:200@patton.com”. The registration messages and incoming calls to the registered contact *must* use the secure transport protocol TLS.

Mode: Registration outbound

Step	Command	Purpose
1	<code>node(regout)# uri-scheme { sip sips }</code>	Configures the URI scheme for outbound registrations and the corresponding calls toward the registered contact. Default is sip.

URI Scheme for Re-routed Calls

It is possible to specify the behavior regarding URI scheme for rerouted calls. This is especially valuable when the call is rerouted to a URI with a different URI scheme than the original URI of the call.

- Rerouted
- Original

Rerouted. The URI scheme is taken from the reroute operation. This means it could be degradation or an enhancement of security of the call. Calls with a SIP URI could be degraded to a SIP URI or calls with a SIP URI could be changed to a SIP URI. This is the default behavior.

Original. The URI scheme for the rerouted call remains the same as for the original call. This means the rerouting operation cannot have an impact on the security of the call. Neither degradation or an enhancement. There is the possibility that the target destination is not compatible to the original URI scheme and would need the URI scheme from the reroute operation.

Mode: Interface SIP

Step	Command	Purpose
1	<code>node(if-sip)[name]# call-reroute uri-scheme { rerouted original }</code>	Configures the URI scheme for outgoing calls which are rerouted. Default is rerouted.

URI Scheme for Transferred Calls

For calls it is possible to specify the behavior regarding URI scheme in the case of a transfer of a call. This is especially valuable when the call is transferred to a URI with a different URI scheme than the original URI of the call.

- Referred
- Original

Referred The URI scheme is taken from the REFER request. This means it could be a degradation or an enhancement of security of the call. Calls with SIPs URI could be degraded to SIP URI or calls with SIP URI could be changed to SIPs URI. This is the default behavior.

Original The URI scheme for the transferred call remains the same as for the original call. This means the transfer operation cannot have an impact on the security of the call. Neither degradation or an enhancement. There is the possibility that the target destination is not compatible to the original URI scheme and would need the URI scheme from the transfer operation.

Mode: Interface SIP

Step	Command	Purpose
1	<code>node(if-sip)[name]# call-transfer uri-scheme { referred original }</code>	Configures the URI scheme for calls which are transferred. Default is referred.

Configure Time

The system time must be configured before you can work with certificates, and enabling the PKI and TLS. It is also very important to set the correct offset of the local time with respect to the UTC time, due to the validity of certificates expressed in UTC time. The configuration option that works best is to update the UTC time from a configured SNTP server and to derive the local time by configuring the static offset of the local time zone with respect to UTC. If a SNTP server is not available, you should first configure the offset of the local time zone and afterwards set the local time with the **et clock** command.

Note It is important to also correctly configure the time of remote devices, which are having TLS connections to our device.

Configure Public-Key Infrastructure (PKI)

The commands that are needed to set up the public-key infrastructure are described in detail in The PKI is responsible for creating, importing and exporting keys and certificates, and storing them persistently in flash memory.

A TLS profile (described in the next section) groups security and other relevant configuration required to open a TLS connection. For this purpose, the TLS profile may refer to a private, an own certificate and many trusted certificates. In order to configure and use a TLS profile, you need to create and/or import keys and certificates. Refer to “Public-Key Infrastructure (PKI)” on page 408 for more information.

About TLS Profiles

A TLS profile is a collection of TLS configuration parameters, which are used by a SIP gateway to establish a secure transport connection to a remote device. Therefore each SIP gateway is bound to one TLS profile. If

your device has to use different PKI keys or certificates to talk to different SIP servers, you have to configure multiple SIP gateway instances, each bound to its own TLS profile.

Basic Operation Principle of TLS

The server side of a TLS connection sends a certificate to the client that serves two purposes: encryption and authentication. First, the certificate contains a public key required to securely negotiate a symmetric session key to encrypt SIP signaling messages. Second, the certificate provides message authentication and integrity by identifying the server. The client is able to authenticate the server by verifying that the server's certificate is derived from one of the locally configured trusted certificates. At the same time, the client is able to verify that the TLS connection is actually terminated by the IP address or domain name listed in the *common name* field of the certificate. The server may request an optional certificate from the client, which is also called mutual authentication.

Encryption usually works with the DEFAULT TLS profile, i.e. without configuring any further options. However, for the authentication and integrity check to work, you have to configure the TLS profile to reflect the trust relationship with the servers/domains you want to communicate with, and you have to manually enable authentication for inbound and outbound connections.

For Trinity, you are able to establish different levels of security; you can set up a quick solution that works with almost zero configuration effort, or you can set up a secure trust relationship in the TLS profile. The following scenarios elaborate more on these security options.

TLS without authentication (default behavior)

This scenario requires no effort to install certificates and only a minimal effort to configure the device, but it also provides the lowest level of security. Each TLS-licensed Patton devices comes with a private key and a default self-signed certificate already installed. The **DEFAULT** TLS profile is configured to use this **DEFAULT** private key and **DEFAULT** own certificate to encrypt SIP signaling messages.

In terms of authentication and integrity check, the security provided by the unaltered TLS profile **DEFAULT** is pretty poor, because authentication for incoming and outgoing TLS connections is switched off. As a consequence, when we accept an incoming TLS connection, Trinity accepts any server certificate without checking a trust relationship and without checking whether the remote IP address matches the *common name* field in the server's certificate. If Trinity acts as a TLS server, the **DEFAULT** TLS profile does not force mutual authentication and thus does not check the identity of the connected client.

Thus, the unaltered TLS profile **DEFAULT** may be helpful for a quick start, but for a serious installation, you should configure a proper trust relationship as described below.

TLS authentication with self-signed certificates

In this scenario we increase the level of security provided by TLS compared to the previous zero-configuration scenario. This is achieved by creating a new self-signed certificate that is only valid for the actual IP address or domain name of the Patton device. See “[Public-Key Infrastructure \(PKI\)](#)” on page 408 to learn how to perform the steps described below:

1. Create a new certificate request, which contains as *common name* field with our public IP address or domain name.
2. Assign a **DEFAULT** private key to the request (or, optionally, assign a newly generated private key).
3. Configure the TLS profile to use the new self-signed certificate and the private key.

Note If such a TLS profile is bound to a SIP gateway, we send this self-signed certificate to identify ourselves. The remote TLS endpoints then are able to verify whether this device's IP address or domain name matches the sent certificate. In order for that to be secure, you also have to export the self-signed certificate and import it to all remote devices as a trusted certificate.

If the device authenticates remote devices, it needs to import the certificates of all remote devices and list them as *trusted certificates* in its TLS profile. In addition, it will switch-on incoming and outgoing authentication to verify whether the remote certificate is derived from one of the trusted certificates listed in the TLS profile. The outgoing authentication process verifies that the remote server endpoint's IP address or domain name matches the common name in the received certificate.

TLS Authentication with 3rd Party Signed Certificates

A generic certification chain consists of the certificate C itself, zero to many certificates of intermediate certification authorities CA1, CA2,..., CAn, and the certificate of the root certification authority CAroot.

In such a case, store certificate C and all certificates of the intermediate certification authorities CA1, CA2,..., CAn on the device that needs to be authenticated. The certificate of the root certification authority CAroot, on the other hand, must be stored as trusted certificate on the device that wants to authenticate the other device(s).

TLS Profile Default

When booting a TLS-licensed Patton device for the first time, a TLS profile with the name DEFAULT is created automatically; it is linked to an automatically-created private key DEFAULT and a self-signed certificate DEFAULT. It is configured not to authenticate and can be used to do authentication-free TLS. For enabling authentication and integrity checks, alter the DEFAULT profile or create a new TLS profile (recommended) and to enable inbound and/or outbound authentication. If a TLS profile is not explicitly bound to a SIP gateway, the DEFAULT TLS profile is used.

Configure TLS Profile

This section describes how to configure TLS profiles, which are those instances in your configuration that collect all TLS-related configuration parameters, and which are used by SIP gateways to establish secure transport connections to remote devices.

Creating a TLS profile and enter configuration mode

This procedure describes how to create a TLS profile and enter the TLS profile configuration mode.

Mode: Administrator exec

Step	Command	Purpose
1	node(cfg)#profile tls <i>name</i>	Creates the TLS profile <i>name</i> and enters its configuration mode.

The parameter *name* is the identifier by which the profile will be known. Entering this command puts you into the TLS profile configuration mode where you can enter the individual TLS configuration parameters.

Use the **no** form of this command to delete a TLS profile. You cannot delete a TLS profile if it is currently bound to a SIP gateway.

To modify a TLS profile that is currently being used by one or more active SIP gateways, the gateways are restarted gracefully; ongoing calls over a certain gateway still use the old TLS configuration until all calls over that gateway have terminated. At that point in time the gateway restarts and all future calls will use the new TLS configuration.

Example: Create a TLS profile

In the following example the TLS profile named *Y_TLS* has been created.

```
node>enable
node#configure
node(cfg)#profile tls MY_TLS
node(pf-tls)[MY_TLS]#
```

Private key

A private key file actually contains a private/public key pair. TLS requires them to secure its key exchange. Thus, it is important to link a TLS profile to a valid private key, which is done by the following command:

Mode: Profile TLS

Step	Command	Purpose
1	node(pf-tls)[name]# [no] private-key pki:private-key/key	Configures the private (and public) key to be used for the TLS key-exchange encryption. This configuration is mandatory!

Note If your TLS profile has a configured chain of *own-certificates* it is required that the first certificate in that chain has been created from a certification request, which contains the corresponding public key.

Example: Create a new private key and link it to a new TLS profile.

In the following example, we let Trinity generate a new private key. We then create a new TLS profile, called *Y_TLS* and link it to the generated private key.

```
node>enable
node#generate pki:private-key/MY_PRIVATE_KEY
node#configurenode(cfg)#profile tls MY_TLS
node(pf-tls)[MY_TLS]#private-key pki:private-key/MY_PRIVATE_KEY
```

Own-certificate chain

Our own certificate is used to identify our device to others when making or accepting TLS connections. The chain of own certificates consists of an order list of one or more links to stored certificates. The first certificate in the list (index 1) is used to communicate our public key to remote TLS devices. It is required that the private key linked by the TLS profile corresponds to the public key in this certificate. Therefore it is mandatory to configure a link to a private key on the TLS profile.

If you want to use a self-signed certificate, the chain of own certificates only contains one entry – a link to that certificate. Otherwise, if your certificate is signed by one or more levels of certification authorities, those intermediate certificates have to follow in the chain. That is, the next entry with index 2 links to the certificate of the certification authority that signed our own certificate at index 1. Then, the entry with index 3 links to the

certificate of the next higher-level certification authority, and so on. The certificate of the root certification authority is not required and may be omitted or may be configured as the final entry in the list.

Mode: Profile tls

Step	Command	Purpose
1	node(pf-tls)[name]# [no] own-certificate [index] pki: own-certificate /certificate	Configures the chain of own certificates. Inserts the certificate at the specified index into the chain or appends it at the end if no index is specified. The sequence of own certificates is of importance! This configuration is mandatory.

Example: Configure a chain of own certificates.

In the following example, we assume that the own-certificate as well as all intermediate certificates that signed our own certificate already have been imported (e.g. with the **own** command). We configure our TLS profile with the certificate chain that represents our own certificate.

```
node(cfg)#profile tls MY_TLS
node(pf-tls)[MY_TLS]#own-certificate 1 pki:own-certificate/MY_CERTn
ode(pf-tls)[MY_TLS]#own-certificate 2 pki:own-certificate/CERT_OF_MY_CA
node(pf-tls)[MY_TLS]#own-certificate 3 pki:own-certificate/CERT_OF_MY_CAS_CA
```

Trusted Certificates

A set of trusted certificates is required to express the trust relationship to other devices. When our device wants to authenticate other devices our TLS profile must be configured with the certificates of all corresponding trusted root certification authorities. In this case a set of one or more trusted certificates is needed. However, when our device doesn't need to authenticate other devices no trusted certificates are needed.

As a default behavior all imported trusted certificates are considered trusted when not configured otherwise in the TLS profile. That is, trusted root certificates usually only have to be imported (e.g. with the **own** command) but must not be linked explicitly in the TLS profile.

Note Certificates stored as **own** certificates are not automatically considered as trusted certificates. If you want to use an own certificate as a trusted certificate as well, you have to copy it to the trusted-certificate section.

Mode: Profile tls

Step	Command	Purpose
1	node(pf-tls)[name]# trusted-certificate all-stored	Configures to use all imported trusted certificates for the current TLS profile. This is the default setting.

In the case that you want to create different TLS profile where each of them links to a different set of trusted certificates, you have to configure those links explicitly.

Mode: Profile tls

Step	Command	Purpose
1	node(pf-tls)[name]# [no] trusted-certificate pki:trusted-certificate/certificate	Adds or removes a certificate to or from the set of trusted certificates of the current PKI profile. Removing the last link sets the TLS-profile configuration back to the default setting, which is to use all trusted certificates in PKI.

Example: Configures a set of trusted profiles.

In the following example, we want to limit the set of trusted devices to those that have their certificate derived from one of the following root certificates.

```
node(cfg)#profile tls MY_TLS
node(pf-tls)[MY_TLS]#trusted-certificate pki:trusted-certificate/ROOT_CA_COMPANY_A
node(pf-tls)[MY_TLS]#trusted-certificate pki:trusted-certificate/ROOT_CA_COMPANY_B
node(pf-tls)[MY_TLS]#trusted-certificate pki:trusted-certificate/ROOT_CA_OUR_OWN_-
COMPANY
```

Authentication

TLS-licensed Patton devices can be configured whether or not to authenticate other devices when establishing TLS connections. If TLS authentication is to be enabled, it is required to have at least one trusted certificate installed on the device and to link the TLS profile to that certificate accordingly (see the description of the **trusted-certificate** command above).

Note This device can only authenticate other devices, but it cannot force other devices to authenticate us. Therefore mutual TLS authentication depends on having each device configured to authenticate the other devices.

Although it is technically possible to authenticate only TLS connections where we are the server side or to authenticate only TLS connections where we are the client side it is not recommended to do so, because in a SIP call, the client and server role may change during a call. (For example, we are the client side for an outgoing call, but the server side if the call is terminated by the remote party.)

Furthermore, having a different configuration for incoming and outgoing TLS connections opens the security hole of abusing connection-reuse for getting or receiving calls to or from unauthenticated remote devices.

Mode: Profile TLS

Step	Command	Purpose
1	node(pf-tls)[name]# [no] authentication incoming	Configures whether or not we are authenticating the remote client in a new TLS connection. This means that we are acting as TLS server for that connection. If enabled, we verify that the received certificate is signed by one of the trusted certificates listed in this TLS profile. Default is enabled.

Step	Command	Purpose
2	node(pf-tls)[name][name]# [no] authentication outgoing	Configures whether or not we are authenticating the remote server in a new TLS connection. This means that we are acting as TLS client for that connection. If enabled, we verify: <ul style="list-style-type: none"> • The received certificate is signed by one of the trusted certificates listed in this TLS profile • The <i>common name</i> field of the received certificate matches the remote IP address or DNS name of the remote TLS endpoint Default is enabled.

Cipher Suites

TLS automatically negotiates to use the crypto algorithms that provide the strongest security, meaning that the remote endpoint may lower the level of security by only providing a weaker crypto algorithm. In high-security environments it may be desirable to restrict the set of offered crypto algorithms. This can be done with the **cipher-suites** command.

Instead of listing each algorithm individually, Trinity offers a selection of pre-configured cipher-suites. Either one or multiple suites can be selected. The suits themselves may overlap with other cipher-suites. The available suites are:

- **ALL** All cipher suites except the eNULL cipher. This is the default setting.
- **DEFAULT** A pre-selected group of ciphers that offer medium to high level of security and at the same time maximum interoperability in most scenarios.
- **HIGH** ciphers with a key-size of 128 bit or more.
- **MEDIUM** ciphers with a key-size of 128 bit.
- **LOW** ciphers with a key-size of 64 or 56 bit.
- **SHA1** ciphers using SHA-1 as digest algorithm.
- **MD5** ciphers using MD5 as digest algorithm.
- **aRSA** ciphers offering RSA authentication.
- **kRSA** ciphers offering RSA key exchange.
- **eNULL** ciphers offering no encryption. Not recommended, use only for debugging.

Mode: Profile tls

Step	Command	Purpose
1	node(pf-tls)[name]# cipher-suites (ALL { DEFAULT HIGH MEDIUM LOW SHA1 MD5 aRSA kRSA eNULL }*)	Configures one or multiple groups of cipher-suites to offer or accept. Default is ALL.

Compression

Compression has the benefit of reducing the size of the TLS packets and thus reducing the bandwidth needed for signaling over TLS. But it could have a negative effect on the performance for encrypting and decrypting the packet and therefore the load of SIP signaling traffic which can be handled on the device.

Mode: Profile TLS

Step	Command	Purpose
1	node(pf-tls)[name]# [no] compression	Configures if we are offering or accepting compression for TLS messages. Default is disabled.

Protocol

The Patton device supports SSL version 3 and its successor TLS version 1. The offering or acceptance can be configured separately for both of them. At least one of the Protocols need to be enabled for TLS operation.

Mode: Profile TLS

Step	Command	Purpose
1	node(pf-tls)[name]# [no] protocol tls-v1	Configures if we are offering or accepting the TLS protocol version 1. Default is enabled.
2	node(pf-tls)[name]# [no] protocol ssl-v3	Configures if we are offering or accepting the SSL protocol version 3. Default is enabled.

Configure TLS profile usage

To actually use TLS for SIP calls and/or registrations, a TLS profile must be bound to a SIP gateway. This allows to quickly changing between different TLS profiles. The DEFAULT profile may be bound if using the default private key and self-signed certificate is fine. Otherwise, if you want to configure a proper trust relationship or use different TLS profiles on different SIP gateways, you have to create new TLS profiles and use them on the respective SIP gateway with the **se** command:

Mode: Context sip-gateway

Step	Command	Purpose
1	node(sip-gw)[name]# use profile tls <name>	Configures which TLS profile to use for connections from and to this SIP gateway. Default is the TLS profile DEFAULT.

Configure TLS transport

Some TLS-related parameters have to be configured outside the TLS profile. In order for the TLS protocol to be used on a SIP gateway, you have to enable it and specify the TCP port over which TLS connections are accepted (port 5061 by default).

It is possible to configure the availability of UDP/TCP, and TLS separately on the transport interface of the SIP gateway. This for example allows a secure configuration where both UDP and TCP are disabled, whereas TLS is enabled as the only transport protocol.

Note In order to use the SIPs URI scheme, TLS must be enabled on the transport interface of the SIP gateway.

Mode: interface sip-gateway

Step	Command	Purpose
1	node(sip-if)[name]# [no] bind ipaddress [<context>] <interface> <label>	Binds the transport interface to a specific IP address on an IP interface in the selected context.
2	node(sip-if)[name]# [no] transport-protocol udp+tcp [<port>]	Enables the reception and sending of SIP requests over UDP and TCP from and to the specified port. Default is enabled. Default port is 5060.
3	node(sip-if)[name]# [no] transport-protocol tls [<port>]	Enables the reception and sending of SIP requests over TLS from and to the specified port. Default is disabled. Default port is 5061.

Configure transport enforcement or fallback

Having multiple transport protocols available to choose from when sending a SIP message makes it desirable to select the order in which they are being used, or even the possibility to force one specific transport protocol. The transport enforcement or fallback configuration only has an effect to outgoing requests to the next hop device. It is not possible to prefer a transport protocol for incoming requests, but TLS can be forced by disabling UDP and TCP on the transport level altogether. Before you are able to force a transport protocol you have to enable the protocol on the transport level (see above). Otherwise requests cannot be sent over the protocol at all.

Note You are able to configure the device to only use SRTP when its session keys can be exchanged over a secure signaling connection. Forcing TLS as transport protocol here is not enough for SRTP to recognize a secure connection. The only way to actually prove SRTP that it is using a secure connection is to configure the SIPs URI scheme. By setting the URI scheme to SIPs, we also force the transport protocol to be TLS, too, but in addition, the transport protocol for SIP signaling messages is forced to be secure beyond the next-hop device. The same applies for having TLS as the preferred transport protocol, where additionally a local fallback to an insecure transport protocol could happen.

The command for selecting a preferred transport protocol allows you to select as a second choice the first fallback and as a third choice the second fallback protocol. As there is a maximum of three protocols the second fallback is implicitly given by choosing the preferred and the first fallback.

Note Not specifying the fallback protocols doesn't disable them, but the order is derived from the internal default ordering. Specifying the transport protocols here does not enable them if they are disabled on the gateway's transport level. The preference of transport has no influence on calls with the URI scheme SIPs. Such calls are always forced to be signaled over TLS. The default ordering of preference is UDP, TCP, and TLS.

Mode: Call Outbound

Step	Command	Purpose
1	node(callout)# [no] transport-protocol force { udp tcp tls }	Forces outgoing INVITE requests to be sent over UDP, TCP, or TLS. If the according transport is not enabled on the sip-gateway level or the remote device does not support it there is no fallback to another transport.
2	node(callout)# [no] transport-protocol prefer { udp tcp tls } *	Sets the preferred transport protocol to choose if multiple are available for an INVITE request. Also specifies to which first and second alternative transport protocol we fall back. Default is to give preference to UDP, and to use TCP and then TLS as fallback.

The same protocol selection can be configured for outbound registrations as well by enforcing or preferring the transport protocol for REGISTER request. The same mechanism is applied for outbound registrations (REGISTER requests) as for outbound calls (INVITE requests).

In addition, we are able to control the transport protocol of incoming calls by the outbound registration; the registered contact address is stored on the server and remembers the transport protocol that was used to register the address. The server then uses this transport protocol to make calls, which the device will receive incoming calls for that registration with the transport protocol the registration was made with.

Mode: Registration outbound

Step	Command	Purpose
1	node(regout)# [no] transport-protocol force { udp tcp tls }	Forces outgoing REGISTER requests to be sent over UDP, TCP, or TLS. If the according transport is not enabled on the sip-gateway level or the remote device does not support it there is no fallback to another transport.
2	node(regout)# [no] transport-protocol prefer { udp tcp tls } *	Set the preferred transport protocol to choose if multiple are available for a REGISTER request. Also specifies to which first and second alternative transport protocol we fall back. Default is to give preference to UDP, and to use TCP and then TLS as fallback.

Configure Non-Default TLS Port Usage

Several SIP-related commands select the remote endpoint's host name and port number. All of them were extended for TLS and now allow you to optionally configure the remote TLS port next to the UDP/TCP port.

For most of these commands the port can be omitted because the standard SIP UDP and TCP port 5060 is used. For cases where the remote device does not use the standard port, the user must specify in the command. Now with the addition of TLS and the possibility to have a transport protocol fallback from TLS to UDP or TCP and vice versa, we need the ability to specify two ports: one for UDP and TCP and the other for TLS.

For all the following commands the same rule applies; the optional port parameter right after the ip-address parameter specifies the port number for UDP and TCP. If not indicated, the port number defaults to 5060. However, this first port parameter is never used for TLS, even if it is the only port specified explicitly, or if UDP and TCP transports are not available.

An additional parameter optionally configures the TLS port. If not indicated, the port defaults to 5061. This parameter is never used for UDP or TCP, even if it is the only port specified explicitly, or if the TLS transport is not be available.

Mode: Interface SIP

Step	Command	Purpose
1	node(if-sip)[name]# [no] remote <host> [<port>] [tls-port <tls-port>]	The existing remote command has the ability to specify a non-default UDP and TCP port and to specify a non-default TLS port. Default UDP/TCP port is 5060. Default TLS port is 5061.
2	node(if-sip)[name]# [no] address-translation outgoing to-header host-part fix <host> [<port>] [tls-port <tls-port>]	The existing address translation to-header command has the ability to specify a non-default UDP and TCP port and to specify a non-default TLS port. Default UDP/TCP port is 5060. Default TLS port is 5061.
3	node(if-sip)[name]# [no] address-translation outgoing request-uri host-part fix <host> [<port>] [tls-port <tls-port>]	The existing address translation Request-URI command has the ability to specify a non-default UDP and TCP port and to specify a non-default TLS port. Default UDP/TCP port is 5060. Default TLS port is 5061.

Mode: Service sip-conference

Step	Command	Purpose
1	node(svc-ls)[name]# [no] conference-server <host> [<port>] [tls-port <tls-port>]	The existing conference-server command has the ability to specify a non-default UDP and TCP port and to specify a non-default TLS port. Default UDP/TCP port is 5060. Default TLS port is 5061.

Mode: Call Outbound

Step	Command	Purpose
1	node(callout)# [no] force-destination <host> [<port>] [tls-port <tls-port>]	The existing force-destination command has the ability to specify a non-default UDP and TCP port and to specify a non-default TLS port. Default UDP/TCP port is 5060. Default TLS port is 5061.
2	node(callout)# [no] proxy <host> [<port>] [tls-port <tls-port>]	The existing proxy command has the ability to specify a non-default UDP and TCP port and to specify a non-default TLS port. Default UDP/TCP port is 5060. Default TLS port is 5061.

Mode: Registration Outbound

Step	Command	Purpose
1	node(regout)# [no] registrar <host> [<port>] [tls-port <tls-port>]	The existing registrar command has the ability to specify a non-default UDP and TCP port and to specify a non-default TLS port. Default UDP/TCP port is 5060. Default TLS port is 5061.
2	node(regout)# [no] force-destination <host> [<port>] [tls-port <tls-port>]	The existing force-destination command has the ability to specify a non-default UDP and TCP port and to specify a non-default TLS port. Default UDP/TCP port is 5060. Default TLS port is 5061.
3	node(regout)# [no] proxy <host> [<port>] [tls-port <tls-port>]	The existing proxy command has the ability to specify a non-default UDP and TCP port and to specify a non-default TLS port. Default UDP/TCP port is 5060. Default TLS port is 5061.

Mode: Message Inbound

Step	Command	Purpose
1	node(msgin)# [no] message-server <host> [<port>] [tls-port <tls-port>]	The existing message-server command has the ability to specify a non-default UDP and TCP port and to specify a non-default TLS port. Default UDP/TCP port is 5060. Default TLS port is 5061.

Configure SRTP

After having configured a secure URI scheme and a secure transport protocols for SIP signaling, you should consider securing the actual media-streams that carry voice or data by using SRTP. For more details, see [“Secure Real-Time Protocol \(SRTP\) Configuration”](#) on page 734.

Security consideration: Why you should use mutual TLS authentication

In the TLS profile, you are able to enable inbound and outbound TLS authentication separately (commands **authentication inbound** and **authentication outbound** respectively). This configuration affects how TLS connections are authenticated, i.e., whether the Patton device verifies TLS server certificates (for outgoing TLS connections) and/or whether it requests and verifies TLS client certificates (for incoming TLS connections). In this section, we will explain why you should enable TLS authentication in both directions for maximum security.

SIP gateway tries to re-use transport connections whenever possible; this is a desired and intended behavior according to the SIP standard. For example, if the Patton device opens ten calls to the exact same SIP server, it would be a waste of resources to establish a new TLS connection for each call. However, since each TLS connection requires CPU resources and network bandwidth to generate keys and negotiate security options, respectively.

The TLS authentication procedure, i.e. the verification of certificates, is only performed when the TLS connection is established. Since TLS connections can be re-used for SIP requests in both directions, not every single SIP call is authenticated. Furthermore, a TLS connection may be used to make calls (or other requests) in the opposite direction, which potentially cancels the intent of your authentication configuration.

IP does not Re-use Connections for Requests in Different Directions

A TLS connection may be opened in either direction during a call: one connection from the calling to the called party, and another one from the called to the calling party. [Figure 107](#) depicts a common registration and call scenario without (left) and with (right) mutual TLS authentication.

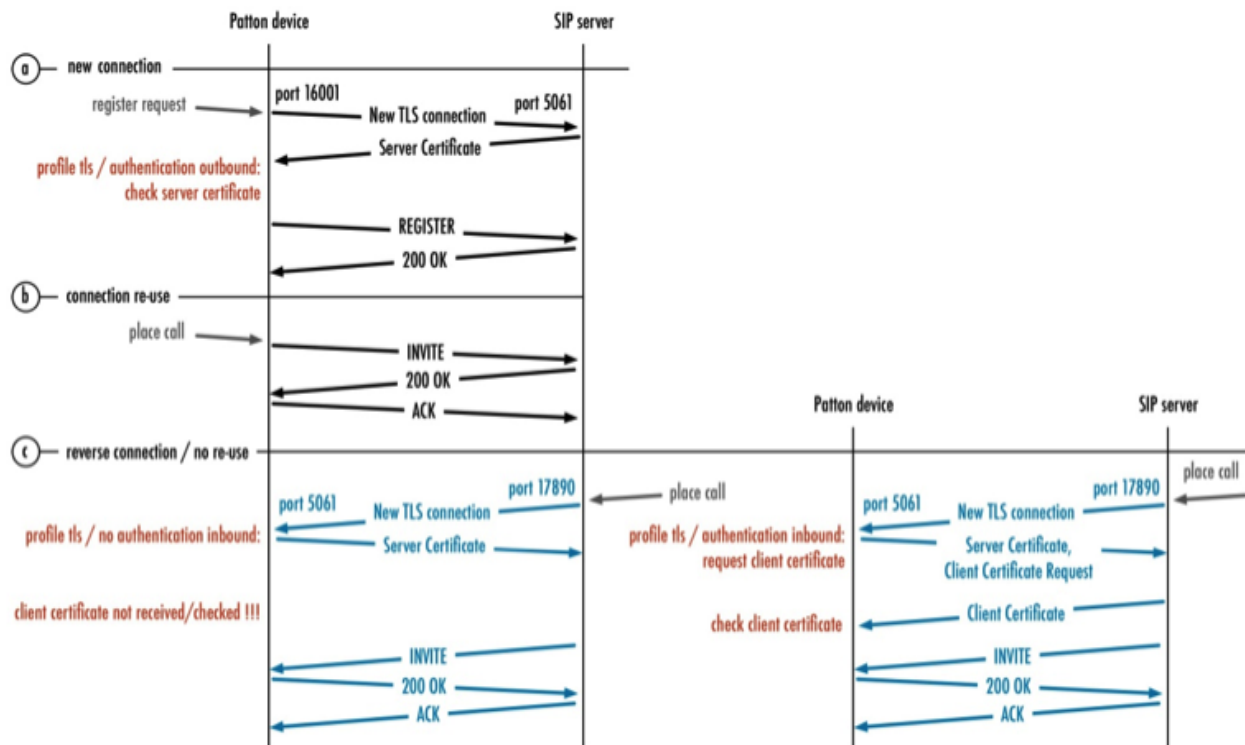


Figure 107. TLS Connection Establishments and Re-use in SIP

- A. When the Patton device has to send a request such as a REGISTER, it may open a new TLS connection. Acting as a TLS client in this scenario, the Patton device uses an ephemeral port (e.g. 16001) and makes a connection to the default TLS port 5061 of the server (or any other port configured in the Request URI). The server then sends back its certificate. The authentication outbound command in the TLS profile decides whether or not our device verifies this certificate (i.e., whether or not it (i) compares the IP address or hostname of the server to the *canonical name* field of the certificate and (ii) verifies that the TLS profile lists in the list of trusted certificates the root CA of the received certificate of the server. Once the TLS handshake completed successfully, the REGISTER request and the corresponding response are sent over this established connection – no new TLS connection has to be opened for a response.
- B. If the Patton device wants to send another request (such as an INVITE or re-REGISTER request) to the same SIP server, it re-uses the TLS connection and therefore does not verify the server's certificate again.
- C. If the SIP server attempts to establish a call to the Patton device at a later point in time. The SIP server cannot re-use the connection, because the Request URI of the INVITE request requires the call to be connected to our local port 5061. Thus the SIP server creates a new TLS connection (blue). The Patton device is the server side for this TLS connection and the authentication inbound command in the TLS profile decides whether or not we request and verify the certificate of the TLS client (which is the SIP server in this case). Note that such a new connection is not only opened for a new call, but also for a new request during an existing call, such as a re-INVITE or a BYE request to renegotiate or drop the call, respectively.

Two Scenarios of TLS Authentication Vulnerability

In the above scenario, assume that **authentication outbound** is enabled but **no authentication inbound** has been configured (left part of figure 107). Let us focus on scenario c where the SIP server places a call towards our device. The SIP server certificate (= TLS client) is not verified since it could be a security issue.

Another, opposite scenario is to have enabled **authentication inbound** but **no authentication outbound** configured. For scenario C, where the SIP server places a call towards our device, we would verify the certificate of the SIP server (=TLS client). The SIP server certificate is no verified again if the server did not act according to the SIP standard and re-used an existing TLS connection we established before. Remember that the command **no authentication outbound** does not check the certificate for outbound TLS connections.

The recommendation is enabling both **authentication outbound** and **authentication inbound** together on the TLS profile whenever possible. The reason for separating the commands was to maximize compatibility in networks where the client does not have its own certificate.

The table below summarizes the vulnerability of the installation due to TLS connections being re-used according to the SIP standard:

Authentication Incoming	Authentication Outgoing	Vulnerability
Enable	Enable	None
Disable	Enable	Connection re-use outgoing
Enable	Disable	Connection re-use oncoming
Disable	Disable	Intentional no authentication

Conditions for the Patton device to Re-use Connections

In order to be precise and complete, this sub-section defines the exact behavior of our SIP application in terms of re-using TLS connections:

- A new outgoing TLS outgoing is opened in the following case:
- SIP Request of a new SIP transaction (e.g. a re-INVITE or BYE request) needs to be sent over TLS, and
- There isn't an opened TLS connection that matches the following parameters:
 - The remote IP address of the connection is equal to the destination IP address of the request,
 - The remote port of the connection is equal to the destination port of the request, and
 - The local IP address of the connection is equal to the bound IP address of the gateway where the request is to be sent.

An existing TLS connection is re-used in the following case:

- The device sends an answer to a incoming request over TLS, or
- The device sends a request over TLS that belongs to an incoming or outgoing transaction, or
- The device sends a SIP Request for a new SIP transaction (e.g. a re-INVITE or BYE request) if there already exists an open TLS connection that matches the following parameters:
 - The remote IP address of the connection is equal to the destination IP address of the request,
 - The remote port of the connection is equal to the destination port of the request, and
 - The local IP address of the connection is equal to the bound IP address of the gateway where the request is to be sent.

Caveat for TLS Profile

The TLS profiles have a limitation in distinctive usage. In certain circumstances, a wrong TLS profile is used for an outgoing connection.

The condition in which this issue was observed is:

- Having multiple SIP gateways with different TLS profiles
- The SIP gateways are bound to the same IP address
- A transaction-forming request needs to be sent over one of these gateways

In all other cases there is no issue at all, especially when one of these conditions is met:

- Having only one gateway
- Using the same TLS profile on all gateways
- Having all gateways bound to different IP addresses
- The TLS connection is opened in incoming direction

We plan to remove this limitation in a future software version.

Showing TLS profile

For debugging purposes, there are two commands that can be entered to take a closer look at the TLS profile internals.

Mode: Administrator executable

Step	Command	Purpose
1	node# show profile tls	Shows a list of all TLS profiles.
2	node# show profile tls <name>	Shows the specific TLS Profiles ready to be used in SIP.
3	node# show profile sip-tls	Shows a list of all TLS profiles ready to be used in SIP.
4	node# show profile sip-tls <name>	Shows the specific TLS profiles ready to be used in SIP, separated into client side and server side.

Troubleshooting

Listed below are common problems that will require a troubleshooting procedure:

- check license
- check time
- check certificates
- check TLS profile status
- check sip-gateway status
- check server name

Check License

In order to use TLS or SRTP for SIP a license has to be installed on the Patton device: “sip-tls-srtp”. This can be verified by showing all currently installed licenses by executing **show system licenses**. Verify that “IP TLS and SRTP” line is displayed.

```
node# show system licenses
Software Licenses
=====
SIP TLS and SRTP
```

Check time

The time settings can be verified by executing the command **show clock**. It is important to have set the correct time for using TLS. The interesting entry here is “UTC Time”, and the entry “Default Offset” must be set correctly.

```
node# show clock
System Time Information
=====
DST Enabled      : false
Default Offset   : +02:00
Local Time       : Monday September 09 2013 08:30:02
UTC Time         : Monday September 09 2013 06:30:02
Clock            : 2013-09-09T08:30:02
```

An example of how to set the correct time:

```
node (cfg)# clock local default-offset +02:00
node (cfg)# clock set 2013-09-09T08:30:00
node (cfg)#
node (cfg)# show clock
System Time Information
=====
DST Enabled      : false
Default Offset   : +02:00
Local Time       : Monday September 09 2013 08:30:02
UTC Time         : Monday September 09 2013 06:30:02
Clock            : 2013-09-09T08:30:02
```

Check Certificates

he validity period of certificates can be examined by **show pki:own-certificates/<name>** or how **pki:trusted-certificates/<name>**

```
node# show pki:own-certificate/MY_CERT
Certificate:
  Data:
    Version: 1 (0x0)
    Serial Number:
      f8:9e:12:de:f9:a9:07:5d
    Signature Algorithm: sha1WithRSAEncryption
    Issuer: L=Bern, CN=192.168.10.10
    Validity
      Not Before: Jul 12 20:48:49 2013 GMT
      Not After : Jul  3 20:48:49 2014 GMT
    Subject: L=Bern, CN=192.168. 10.10
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      Public-Key: (512 bit)
      Modulus:
        00:e8:f8:07:3c:77:b4:c3:37:d6:58:11:66:fb:3d:
        82:52:bf:5c:b3:d9:53:e8:eb:9b:3e:8a:e6:23:ab:
        c7:b3:f0:6c:12:e3:97:63:42:ab:86:f2:13:f2:f1:
        d7:24:04:af:a1:0e:3d:22:03:a9:60:73:aa:d4:cd:
        70:82:50:08:55
      Exponent: 65537 (0x10001)
    Signature Algorithm: sha1WithRSAEncryption
      7e:ab:68:fb:d1:50:ce:5e:69:80:85:1f:73:84:ba:03:6a:76:
      6e:a9:ad:e3:27:08:67:cd:4d:7b:fe:51:56:4b:50:9b:37:90:
      25:f0:a2:78:73:33:84:31:09:08:75:b0:18:ad:7e:bc:8d:f5:
      8d:6b:60:20:32:61:ac:c6:10:9d
```

Check TLS Profile Status

The loading status of the TLS profile as it is seen by our SIP application can be examined by executing the command **show profile sip-tls <name>**. It is important here to check that the “Client status” is shown as “Client successful configured” and the “Server status” reported as “Server successful configured”.

If that is the case the following internal checks have been passed successfully:

- The loaded certificates are syntactically valid
- The dependency of the own-certificate chain is correct
- The own-certificate has really been derived from the configured private key

In addition there is a line for each certificate with the status of the validity period, which should be shown as “OK”. The system time is within the validity period of that certificates at showing time.

```
node# show profile sip-tls DEFAULT

SIP-TLS Profile: DEFAULT
=====

Used:                               by 1 module(s)
Client:                              enabled
Client status:                       Client successful configured
Server:                              enabled
Server status:                       Server successful configured

Client Context:
-----

TLS V1 Protocol:                     enabled
SSL V3 Protocol:                     enabled
Compression:                         enabled
Peer Authentication:                 disabled
Ciphers:                             ALL
Validity of Own Certificates:        1
  Own Certificate 1:                  OK
Validity of Trusted Certificates:    2
  Trusted Certificate 1:              OK
  Trusted Certificate 2:              OK

Server Context:
-----

TLS V1 Protocol:                     enabled
SSL V3 Protocol:                     enabled
Compression:                         enabled
Peer Authentication:                 disabled
Ciphers:                             ALL
Validity of Own Certificates:        1
  Own Certificate 1:                  OK
Validity of Trusted Certificates:    2
```

Check SIP Gateway Status

The status of the SIP gateway can be examined by executing the command **show context sip-gateway detail 4**. Note the “Used TLS profile” for TLS and the status of the transport binding TLS.

```
node # show context sip-gateway detail 4

Gateway: GW_SIP
=====

Active:                               yes
Number of connected calls:             0
Number of ongoing calls:               0
Accumulated number of calls:           0
Used TLS profile:                       DEFAULT
Bound Location Service:                 PATTON

SocketInterface: GW_SIP.MAN
-----

State:                                 Opened

Transport binding:                      UDP
State:                                  Opened
IP-Address:                             192.168.10.10
Port:                                    5060

Transport binding:                      TCP
State:                                  Opened
IP-Address:                             192.168.10.10
Port:                                    5060

Transport binding:                      TLS
State:                                  Opened
IP-Address:                             192.168.10.10
Port:                                    5061

.....continuing
```

Check Server Name

When authenticating outbound while the TLS connection is opened from this device, the common name in the server certificate must match the host name of the request URI. To verify if this check fails or is successful for the TLS connection creation, the user will need to trace.

```
node # debug all-sip detail 5
```

Execute a test-call to trace the sip-signaling and the server-name check during the TLS handshake. Keywords to look for in the trace are:

- TLS handshaking request arrived
- TLS handshake approval for client connection

```
[DBG] [STACK] TLS handshaking request arrived
# [DBG] [STACK]   bIsClient: yes
# [DBG] [STACK]   bPeerAuth: yes
# [DBG] [STACK]   Number of certificate(s) in the chain: 1
# [DBG] [STACK]   Certificate 0
# [DBG] [STACK]     Version: EVERSION_V3
# [DBG] [STACK]     Serial:
# [DBG] [STACK]     B7 0A E7 84 40 91 2A 2C
# [DBG] [STACK]     Not Before: 2000/6/2 23:40:15
# [DBG] [STACK]     Not After: 2999/10/3 23:40:15
# [DBG] [STACK]     rstrPeerHostname: 172.16.40.7
# [DBG] [STACK] No SubjectAlternateName Extensions in certificate
# [DBG] [STACK] No CommonName in certificate
# [DBG] [STACK] TLS handshake approval for client connection: NOT
APPROVED, PeerHostname does not match CommonName or SubjectAltName
```

There are three possible outcomes:

1. **A handshaking request visible and not approve.** If the TLS handshake request was not approved there wasn't a SubjectAlternateName or CommonName in the server certificate received which matched the PeerHostName.
 - Certificate contains no SubjectAlternateName or CommonName: The servers certificate needs to be created new with specifying at least one of these parameters.
 - Certificate contains a SubjectAlternateName or CommonName which does not match the PeerHostName: change address-translation in SIP call-control interface to match the resulting request URI of the INVITE to the SubjectAlternateName or CommonName.
2. **A handshaking request visible and approved.** If the TLS handshaking request was approved, continue troubleshooting by capturing Wireshark traces of the TLS connection establishment after getting a TLS handshake request. Probably the TLS connection establishment was successful and the failure happens later in normal SIP signaling
3. **No handshaking request visible:** If the section is not visible, continue troubleshooting by capturing Wire-shark traces of the TLS connection establishment before getting a TLS handshake request.

Note If the SubjectAlternateName or CommonName in the certificate is a host name that needs to be resolved through DNS, the PeerHostName needs to be a DNS name too. The same applies for IP-addresses; if the SubjectAlternateName or CommonName in the certificate is an ip-address, the Peer-HostName needs to be an ip-address as well. IP-addresses do not match the DNS names and vice versa.

Chapter 55 TLS Configuration for Lync

Chapter contents

Introduction	709
Configuration options: mutual vs. server authentication	709
Task List.....	709
1. Prepare the Lync server.....	710
Option A: Prepare the Lync server for mutual authentication	710
Lync CA: Create an authentication template for mutual authentication	711
Lync CA: Activate the new template on the Certification Authority (CA)	712
Lync: Issue a certificate for the Lync server with the mutual authentication template	712
Lync: Configure the Lync server	712
Option B: Prepare the Lync server for server authentication only	713
Lync: Issue a certificate for the Lync server with the server authentication template	713
Lync: Configure the Lync server	713
2. Configure basic settings on the Patton device running Trinity.....	713
Trinity: Configure DNS	714
Trinity: Configure time	714
3. Issue a certificate on the Lync server and import it on the Patton device running Trinity.....	714
Option A: Export certificate request from Trinity	714
Trinity: Generate private key	715
Trinity: Generate certificate request	715
Trinity: Export certificate request	716
Lync CA: Sign certificate request	716
Trinity: Import own certificate	718
Trinity: Import trusted certificate	718
Option B: External certificate administration	719
Lync CA: External certificate creation	719
Trinity: Import private key	719
Trinity: Import own certificate	720
Trinity: Import trusted certificate	720
4. Configure TLS on the Patton device running Trinity.....	720
Configure Trinity TLS profile	720
Option A: Configure TLS profile with mutual authentication	720
Option B: Configure TLS profile with server authentication	721
Trinity: Configure SIP with TLS	721
Final Configuration on Trinity device	722

Introduction

This chapter explains how to integrate a Patton VoIP gateway to a Microsoft Lync network. This involves four major steps: (1) Prepare the Lync server, (2) configure basic settings on the Patton device, (3) enroll certificates on the Lync server and import them to the Patton device, and (4) configure Transport Layer Security (TLS) on the Patton device.

Some steps may not be necessary in your set up. For example, step 1 is not required if you have already set up a Lync template to enroll certificates for mutual authentication. Anyway, we discuss all steps in detail for your reference.

Configuration options: mutual vs. server authentication

This chapter is structured along the four major steps to prepare and configure both the Lync server as well as the Patton device. Two of the four steps are broken down into optional task lists (see steps 1 and 3 in the task list below). The basic options you can choose from are to either use Mutual Transport Layer Security (MTLS) or just use TLS for server authentication. For security reason, we recommend to use MTLS.

Both TLS and MTLS provide encrypted SIP signaling channels and endpoint authentication. TLS enables clients to authenticate a server to which they connect. This simple method is used in clientserver architectures such as a web client connecting to a web server. The client opens a TLS connection to the server and receives a certificate from that server. To be valid, the certificate (1) must have been signed by a CA that is also trusted by the client, and (2) the DNS name or IP address of the server must match the Common Name (CN) field of the certificate. If the certificate is valid, the client trusts the server and opens the connection.

In SIP networks, a SIP endpoint is both client and server at the same time. For calls from the Patton gateway to the Lync server, the Patton device is the TLS client whereas the Lync server is the TLS server. However, if the Lync server places a call to the gateway, the Lync server is the TLS client whereas the Patton device is the TLS server. Such server-to-server communication requires MTLS connections for maximum security. With MTLS the TLS server is also able to authenticate the client. Both endpoints exchange certificates from a mutually trusted Certification Authority (CA). These certificates prove the identity of each endpoint to the other.

It is therefore important that the Lync server is able to enroll certificate for MTLS. If the Patton device is provided with a certificate that only supports server authentication, the Patton device will not be able to authenticate the Lync server if it opens a TLS connection itself. This is why we strongly recommend to re-configure the Lync server such that it is able to enroll certificates with both client and server authentication enabled (for MTLS connections).

Task List

Please follow the task lists in each of the four steps to integrate a Patton device into your Lync network:

1. Prepare the Lync server
 - ☞ Option A: Prepare the Lync server for mutual authentication
 - Lync CA: Create an authentication template for mutual authentication
 - Lync CA: Activate the new template on the Certification Authority (CA)
 - Lync: Issue a certificate for the Lync server with the mutual authentication template
 - Lync: Configure the Lync server

- ❖ Option B: Prepare the Lync server for server authentication only
 - Lync: Issue a certificate for the Lync server with the server authentication template
 - Lync: Configure the Lync server
- 2. Configure basic settings on the Patton device running Trinity
 - ❖ Trinity: Configure DNS
 - ❖ Trinity: Configure time
- 3. Issue a certificate on the Lync server and import it on the Patton device running Trinity
 - ❖ Option A: Export the certificate request from Trinity
 - Trinity: Generate private key
 - Trinity: Generate certificate request
 - Trinity: Export certificate request
 - Lync CA: Sign certificate request
 - Trinity: Import own certificate
 - Trinity: Import trusted certificate
 - ❖ Option B: External certificate administration
 - Lync CA: External certificate creation
 - Trinity: Import private key
 - Trinity: Import own certificate
 - Trinity: Import trusted certificate
- 4. Configure TLS on the Patton device running Trinity
 - ❖ Configure Trinity TLS profile
 - Option A: Configure TLS profile with mutual authentication
 - Option B: Configure TLS profile with server authentication
 - ❖ Trinity: Configure SIP with TLS

1. Prepare the Lync server

This step-by-step guide splits at this point. Depending on whether you want to issue certificates with mutual authentication or server authentication only, you have to continue reading the section for Option A or Option B, respectively. After having prepared the Lync server, please continue with step 2.

Option A: Prepare the Lync server for mutual authentication

The following steps are required to create certificates on the Lync server. These certificates will be (a) used to configure the Lync server itself (described here) and (b) imported to the Patton device (described in step 3). Option A describes how to make a template to create certificates with mutual authentication enabled. Mutual

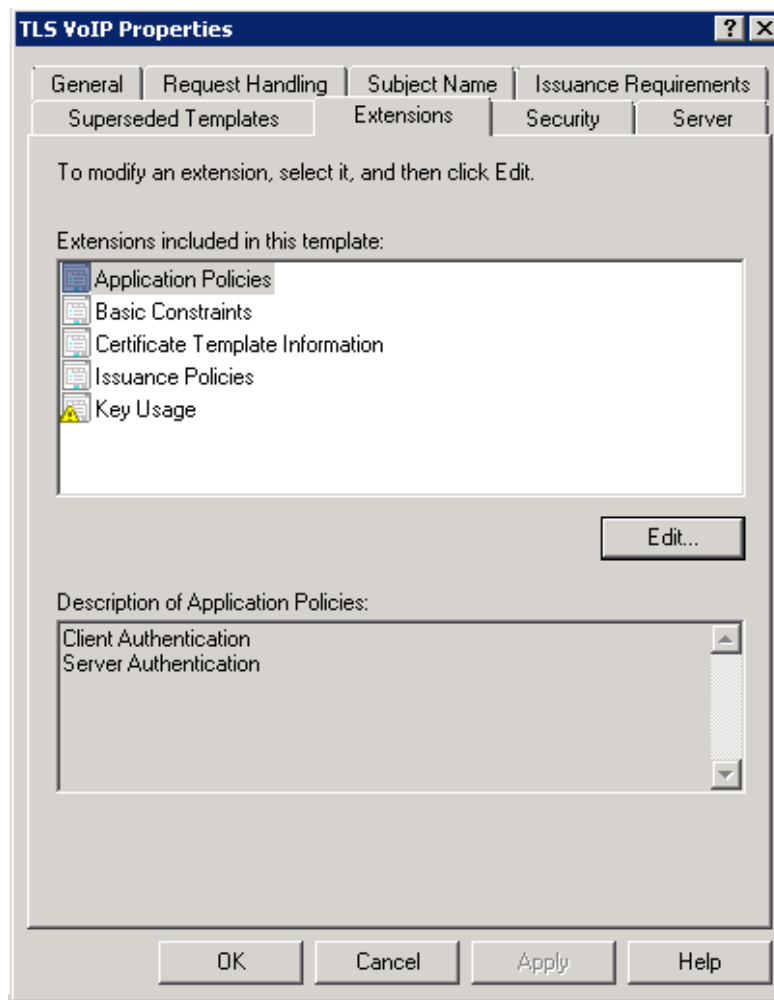
authentication allows the Patton device to authenticate the Lync server in TLS connections no matter whether the Patton device is TLS client (e.g. outgoing SIP call) or TLS server (e.g. incoming SIP call).

Lync CA: Create an authentication template for mutual authentication

On the Domain Controller that is used for certificate enrollment in your Lync network, you have to create a certificate template that can be used to generate certificates for MTLS connections. The template is used in the Lync setup to sign certificates, i.e., all certificates signed with this template allow the Lync server and the Patton device to perform mutual authentication (client authentication and server authentication).

To create this template, perform the following steps on the Domain Controller:

- Install the certificate template snap-in,
- Duplicate the template “Web Server”,
- Give it a new name, e.g. “TLS VoIP”, and
- Add the value “Client authentication” to the “Application Policies” extension as depicted in the figure below.



Lync CA: Activate the new template on the Certification Authority (CA)

The next step is to activate the created authentication template on Lync's Certification Authority (CA). This step is required in order to sign certificates for MTLS. To activate the new authentication template, perform the following steps on the Domain Controller:

- Install the certification authority snap-in,
- Choose *Certificate Templates* -> *New* -> *Certificate Template to Issue*, and
- Select the new Certificate template "TLS VoIP"

Lync: Issue a certificate for the Lync server with the mutual authentication template

With the newly created certificate template you are now able to create a certificate for the Lync server itself. Use the Deployment wizard on the Lync server as follows:

- Open the Lync Server 2013 Deployment wizard
- Request, Install or Assign Certificates -> Run
- Select Default certificate and press Request
- Continue with the wizard until you have reached the step *Specify Alternate Certificate Template*
- **Specify Alternate Certificate Template:** enter the name of the new template "TLVoIP". Important note: You have to enter the name without spaces as in the Template name and not as in the Template display name.
- Continue with the wizard until you have reached the step *SIP Domain setting on Subject Alternative Names*
- **SIP Domain setting on Subject Alternative Names:** Select the configured SIP domains. One of these SIP domains later has to be configured on the Trinity device. Alternatively, you can also configure the Trinity device with one of the Subject Alternative Names in the next pane:
Configure Additional Subject Alternative Names.
- Continue with the wizard until you have reached the step *Online Certificate Request Status*
- Select *Assign this certificate to Lync Server certificate usages*
- *Finish*
- Execute the Certificate Assignment wizard

Note This procedure may be different for your application setup or require more or other parameters to be set. The steps highlighted in bold however are crucial to create a certificate with the MTLS template you have created before.

Lync: Configure the Lync server

The exact procedure of how to configure the Lync server is out of the scope of this document. However, please make sure to enable the following features:

- Use SRTP
- Perform mutual authentication

Option B: Prepare the Lync server for server authentication only

Option B describes how to make a template to create certificates without mutual authentication, which is not recommended for security reasons.

Lync: Issue a certificate for the Lync server with the server authentication template

We use the existing Web Server template to create a certificate for the Lync server itself. Use the Deployment wizard on the Lync server as follows:

- Open the Lync Server 2013 Deployment wizard
- *Request, Install or Assign Certificates -> Run*
- Select *Default certificate* and press *Request*
- Continue with the wizard until you have reached the step *Specify Alternate Certificate Template*
- **Specify Alternate Certificate Template:** enter the name of the template “WebServer”.

Note You have to enter the name without spaces as in the Template name and not as in the Template display name.

- Continue with the wizard until you have reached the step *SIP Domain setting on Subject Alternative Names*
- **SIP Domain setting on Subject Alternative Names:** Select the configured SIP domains. One of these SIP domains later has to be configured on the Trinity device. Alternatively, you can also configure the Trinity device with one of the *Subject Alternative Names* in the next pane:
Configure Additional Subject Alternative Names.
- Continue with the wizard until you have reached the step *Online Certificate Request Status*
- Select *Assign this certificate to Lync Server certificate usages*
- *Finish*
- Execute the Certificate Assignment wizard

Note This procedure may be different for your application setup or require more or other parameters to be set. The steps highlighted in bold typeface however are crucial to create a certificate with the *Web Server* template.

Lync: Configure the Lync server

The exact procedure to configure the Lync server is out of the scope of this document. However, please make sure to enable/disable the following features:

- Use SRTP
- Disable mutual authentication

2. Configure basic settings on the Patton device running Trinity

The next step is to configure basic settings on the Patton device such as a remote DNS server and the system time. Both settings are required for the Patton device to verify certificates. A DNS server is required when the Lync server is configured with its DNS name on the Patton device, to resolve the DNS name to an actual IP address, and the time is required to check whether a certificate is still valid.

Trinity: Configure DNS

To configure an external DNS server (e.g. reachable at 172.16.75.202) enter the following commands in the configuration mode:

```
node(cfg)#dns-server
node(cfg)(dns)#name-server address 172.16.75.202
node(cfg)(dns)#no shutdown
```

Trinity: Configure time

To synchronize the system time with an external NTP server (e.g. swisstime.ethz.ch) enter the following commands in the configure mode:

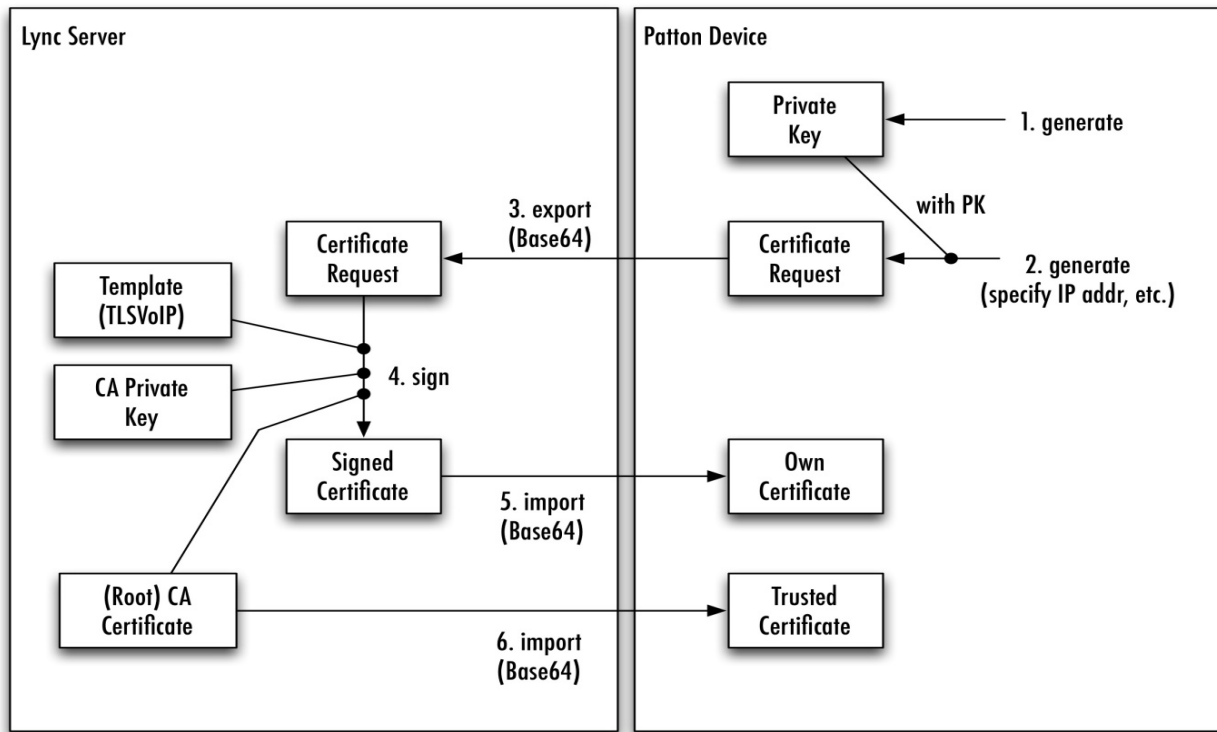
```
node(cfg)# clock local default-offset +02:00
node(ntp)# ntp
node(ntp)# server swisstime.ethz.ch
node(ntp)# no shutdown
```

3. Issue a certificate on the Lync server and import it on the Patton device running Trinity

The next step involves creating certificates for the Patton device on the Lync server. The guide splits again at this point. There are two options how to bring certificates to the Patton device. Option A describes a method where the Patton device creates a certificate request, which then has to be signed by the Lync CA. Option B on the other hand describes a procedure where all files are generated on the Lync server and then imported to the Patton device. Please follow the sub-sections for Option A or Option B, respectively. After having imported the certificates on the Patton device, continue with step 4.

Option A: Export certificate request from Trinity

The figure below depicts the procedure of issuing certificates on the Lync sever and importing them on the Patton device. After (1) creating a private key and (2) a certificate request on the Patton device, you have to (3) export the request and (4) sign it on the Lync server. The signed certificate (5) as well as the certificate of Lync's CA (6) must then be imported back to the Patton device.



Trinity: Generate private key

With Option A, the private key is generated locally on the Patton device. A new private key is needed, which fulfills the requirement of the key length of the CA template. For example, to create a new private key of name *SECRET1*, using a 2048 bit long modulus, execute the following command:

```
node(cfg)#generate pki:private-key/SECRET1 key-length 2048
Generating RSA private key, 2048 bit long modulus
..+++
.....+++
e is 65537 (0x10001)
writing RSA key
```

The generated key is stored in the persistent memory of the Patton device. That is, after a reboot, the key is still available.

Trinity: Generate certificate request

As a next step, you have to generate a new certificate request including the public key that matches the private key generated in the previous step. A certificate request contains the same information fields than are also present in the final certificate (e.g. common name field), but it is not signed by the CA yet.

Enter the following command to generate a certificate request *REQUEST1* with the created private key *SECRET1* for a device with the organization information shown below:

```
node(cfg)#generate pki:certificate-request/REQUEST1 private-key
pki:private-key/SECRET1 country CH state Bern locality Bern organization
```

```
Patton organization-unit RND common-name 172.16.44.111
```

Note The common-name parameter at the end of the command must match the DNS name or IP address of the Patton device. This must be the same name the Lync server uses to reach the gateway.

Trinity: Export certificate request

The Lync server now has to sign the certificate. Please export the certificate request on the Patton device as follows:

```
node(cfg)#export pki:certificate-request/REQUEST1

-----BEGIN CERTIFICATE REQUEST-----
MIICpTCCAY0CAQAwYDELMAkGA1UEBhMCQ0gxDTALBgNVBAGMBEJlcm4xDTALBgNV
BAcMBEJlcm4xDzANBgNVBAoMBlBhdHRvbjEMMAoGA1UECwwDUk5EMRQwEgYDVQQD
DAsxOTIuMTY4LjEuMjCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAR9
TdqGIFBoz2w/d/0vba4Ak6WHFSRf7aS+fp+/0Z/9bFatdKSHxAjPa5J+wpYI9yN
mgI2+5dmAgHg0geCVvFI9wo/YQfgtHGQ1XwxDACA9a6R0urAFcOAtW36HkgUwpBj
7BZQnLqXq9W3u003c5B+wr/ZpUWOZLj5cADbYJtMk6a/cUcjlS4egaESLCWQoYP/
jI7Rey2wBSmUAOpqp46IN4lsq92xv5wT+Z5R1zleJts4RkT/XeV8MQ/g1ZX6nS/U
xB5pfFwrtwLuIT2XNqQ19xrSgee2smi3+8y3j3+dE7PP2rbMHe6UHD8T66HxBvZ4
JtfXoEDTLvVYlOU+pFUCAwEAAaAAMA0GCSqGSIb3DQEBAQUAA4IBAQBFqrYiVrse
F5gOCFj4WeV6cSdaBcU2Jv+AshItCBQ07SOU3cZSMr1OITsIOs/gDkVxDFBEK8XU
YckyQGr/aQWcAfCwdxzngastw5dRTyni0ybdyLFABkhmMbC2zKL2dwWjeeZSee0c
oTQzlc5Q4tUjzzOZQ8CBnRm7QtVoJ6X3OpzeB7I1xFawIzV9g00cPdJfSAq733TD
NF9cuDx4qqSIBIJ9Yv1C2X6T0WjTyOHQDICHAR58PTrT+MzR98LJkPMFX0bBoQd
y3f71W3oPz602akU48nRPPPrToFm4Z1zULiCrGGEhaMQK2bPMxoTt//HC/jCyNe+
wEPaIWE1LmPz
-----END CERTIFICATE REQUEST-----
```

Either copy the printout of the export command including the begin/end commands from the terminal or execute the following command to upload the request to a TFTP server:

```
node(cfg)#copy pki:certificate-request/REQUEST1
tftp://<server>/REQUEST1.txt
```

Lync CA: Sign certificate request

To sign the certificate on the responsible Certification Authority (CA), perform the following steps:

- Open the web GUI of the CA to sign our certificate request: <ip-addr>/certsrv/certrqxt.asp.
- Paste the certificate request content in Saved Request
- For mutual authentication (MTLS): Choose the template TLSVoIP under Certificate Template
- For server authentication: Choose the template Web Server under Certificate Template
- The screen should display the filled-out form as depicted below. Press Submit

Microsoft Active Directory Certificate Services -- new-SRV-LYNC2-CA [Home](#)

Submit a Certificate Request or Renewal Request

To submit a saved request to the CA, paste a base-64-encoded CMC or PKCS #10 certificate request or PKCS #7 renewal request generated by an external source (such as a Web server) in the Saved Request box.

Saved Request:

Base-64-encoded certificate request (CMC or PKCS #10 or PKCS #7):

```

-----BEGIN CERTIFICATE REQUEST-----
MIICpTCCAYOCAQAwYDELMAkGA1UEBhMCQ0gxDTALBgNVBAGMBEUJlcm4xDALBgNV
BACMBEUJlcm4xDzANBgNVBAoMB1BhdHRvbjEMMAoGA1UECwwDUk5EMRQwEgYDVoQD
DAsxOTIuMTY4LjEuMjCCASIdWQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAR9
TdqgGiFBoz2w/d/OvbA4Ak6WHFSRf7aS+fp+/0Z/9bFatdKSHxAjPa5J+wpYI9yN
mgI2+5dmAgHg0geCVvFI9wo/YQfgtHGQ1XwxDACA9a6R0urAFcOAtW36HkgUwpBj
7BZQnLqXq9W3u003c5B+wzr/ZpUWOZLj5cAdbyJtMk6a/cUcjlS4egaESLQWQoYP/
jI7Rey2wBSmUAOpqp46IN41sq92xv5wT+Z5R1z1eJts4RkT/XeV8MQ/g1ZX6nS/U
xB5pfFwrtwLuIT2XNqQ19xrSgee2smi3+8y3j3+dE7FP2rbMHe6UHD8T66HxBvZ4
JtfxoEDTLvVY1oU+pFUCAwEAAAAMA0GCSqGSIb3DQEBAQUAA4IBAQBfqrYiVrse
F5gOCFj4WeV6cSdaBcU2Jv+AshItCBQ07SOU3cZSMz1OITsIO5/gDkVxDFBEK8XU
YckyQGr/aQWcAfCwdxzngastw5dRTyni0ybdyLFABkxhMbc2zKL2dwWjeeZSeeOc
oTQzlc5Q4tUjzzOZQ8CBnRm7QtVoj6X3OpzeB7I1xFawIzV9g00cPdjfSAq733TD
NF9cuDx4qqsSIBIJ9Yv1C2X6TOWjTyOHQDICHAr58PTrT+MzR98LJkPMF0bBoQd
y3f71W3oPz602akU48nRPPPrToFm4Z1zULiCrGGEhaMQK2bPMxoTt//HC/jCyNe+
wEPaIWE1LmPz
-----END CERTIFICATE REQUEST-----

```

Certificate Template:

TLSforVoIP

Additional Attributes:

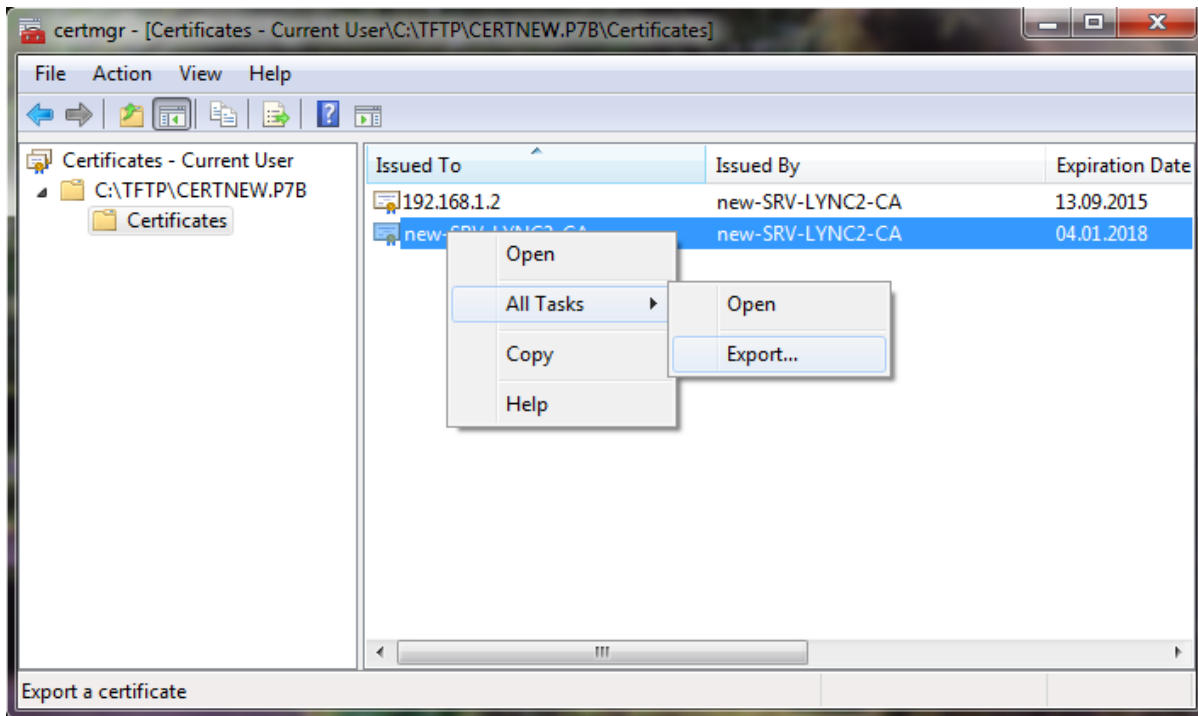
Attributes:

- On the next page click Download certificate chain and save it.
- Open the saved p7b file with a double click (opens certmgr on Windows)
- Note that it is not possible to import whole certification chains to the Trinity device. You have to export each certificate of the chain separately. Thus, for each certificate do the following tasks:
 - Right click -> All Tasks -> Export
 - Select Base64 encoded X.509
 - Save in your TFTP-server folder

In this document, we assume that you give the certificates the following names:

- Name the issued certificate for the Patton device “cert.cer”.
- If they exist for you set up, name the intermediate-certification-authority certificates “ca1.cer”, “ca2.cer”,... , and “caN.cer”. In this example, we assume that there are no intermediate CAs, i.e., that the device certificate is signed directly by the root CA.

- Name the certificate of the Root Certification Authority “caRoot.cer”.



Trinity: Import own certificate

As a next step, you have to import the issued device certificate and potentially all certificates of the intermediate certification authorities to the Patton device. If the device certificate is signed by the root CA directly, you only have to import the device certificate here.

Copy the device certificate from your TFTP server to the Patton device’s persistent memory by executing the following command:

```
node(cfg)#copy tftp://<ip-address>/cert.cer pki:own-certificate/CERT1
```

If you have intermediate certificates, copy them as well with the following commands:

```
node(cfg)#copy tftp://<ip-address>/ca1.cer pki:own-certificate/CA1
node(cfg)#copy tftp://<ip-address>/ca2.cer pki:own-certificate/CA2
```

Trinity: Import trusted certificate

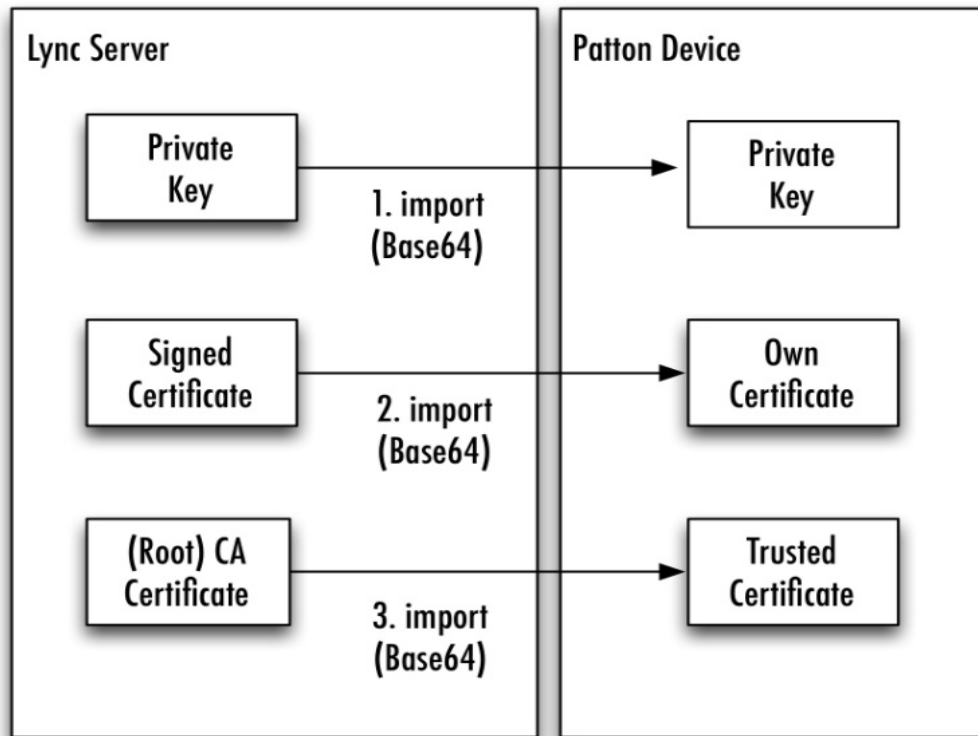
Finally, you have to import the certificate of the Root Certification Authority to the Patton device. We store it as trusted certificate, which means that we trust TLS-received certificates that are directly or indirectly signed by this Root CA certificate. Execute the following command to import the certificate of the Root CA:

```
node(cfg)#copy tftp://<ip-address>/caRoot.cer pki:trusted-certificate/ROOT
```

Note In step 4 you will have to configure which of the imported certificates to use on a specific SIP gateway. This enables multi-homing solutions, where the Patton device is able to participate in multiple SIP networks with different certificates and trust relationships.

Option B: External certificate administration

Instead of generating private key and certificate request on the Patton device, you may generate all keys and certificates offline on the server and import them to the Patton device. The figure below depicts this alternative.

**Lync CA: External certificate creation**

A detailed description of how to create private keys and certificates on an offline CA is out of the scope of this document. Here, we only report briefly on some constraints for using certificates on Trinity devices.

- The Subject Common Name or Subject Alternate Name must be the IP address or host name of the Trinity device.
- The Public Key enclosed in the certificate must be the counterpart of the imported private key .
- Key usage extension must be absent or must contain “Digital Signature” and “Key Encipherment”.
- Extended key usage extension be absent or contain “Server Authentication” and “Client Authentication” for doing MTLS (mutual TLS), or “Server Authentication” for server authentication only.

Trinity: Import private key

As a first step, you have to import the generated private key. This private key must be the counterpart of the public key, which is part of the issued certificate. Execute the following command to import the private key:

```
node(cfg)#copy tftp://<ip-address>/secret.key pki:private-key/SECRET1
```

Trinity: Import own certificate

Next, you have to import the certificate chain for the client, consisting of the issued certificate and all certificates of any intermediate certification authorities. If there are no intermediate CAs only one certificate must be imported here:

```
node(cfg)#copy tftp://<ip-address>/cert.cer pki:own-certificate/CERT1
```

In addition, execute the following command to import the certificates of the optional intermediate CAs:

```
node(cfg)#copy tftp://<ip-address>/ca1.cer pki:own-certificate/CA1
node(cfg)#copy tftp://<ip-address>/ca2.cer pki:own-certificate/CA2
```

Trinity: Import trusted certificate

Finally, the certificate of the root CA must be imported as trusted certificate:

```
node(cfg)#copy tftp://<ip-address>/caRoot.cer pki:trusted-certificate/ROOT
```

4. Configure TLS on the Patton device running Trinity

After issuing and importing private key and certificates, you are ready to complete the Trinity configuration for TLS by creating a TLS profile and binding it to a SIP gateway.

Configure Trinity TLS profile

Depending on whether or not you have created a certificate for mutual authentication, please follow Option A or Option B, respectively.

Option A: Configure TLS profile with mutual authentication

Create a new TLS profile and use the imported private key:

```
node(cfg)#profile tls TLS_AUTH
node(pf-tls)[TLS_AUTH]#private-key pki:private-key/SECRET1
```

Configure the own-certification chain. If there are no intermediate certification authorities there is only one certificate configured here – the certificate of the device itself.

```
node(pf-tls)[TLS_AUTH]#own-certificate 1 pki:own-certificate/CERT1
```

If your certificate has been signed by intermediate CAs, add them to the own-certificate list as well (in the order of signature):

```
node(pf-tls)[TLS_AUTH]#own-certificate 2 pki:own-certificate/CA1
node(pf-tls)[TLS_AUTH]#own-certificate 3 pki:own-certificate/CA2
```

Configure the root CA's the Patton device shall trust. When the Patton device receives a certificate from a peer TLS endpoint, it only renders the certificate valid if it is signed by one of the listed root CAs:

```
node(pf-tls)[TLS_AUTH]#trusted-certificate pki:trusted-certificate/ROOT
```

Configure mutual authentication:

```
node(pf-tls)[TLS_AUTH]#authentication incoming
node(pf-tls)[TLS_AUTH]#authentication outgoing
```

Option B: Configure TLS profile with server authentication

If you don't want to use MTLS, i.e. if the Trinity device shall only authenticate TLS servers, create a new TLS profile and use the imported private key:

```
node(cfg)#profile tls TLS_AUTH
node(pf-tls)[TLS_AUTH]#private-key pki:private-key/SECRET1
```

Configure the own-certification chain. If there are no intermediate certification authorities there is only one certificate configured here – the certificate of the device itself.

```
node(pf-tls)[TLS_AUTH]#own-certificate 1 pki:own-certificate/CERT1
```

If your certificate has been signed by intermediate CAs, add them to the own-certificate list as well (in the order of signature):

```
node(pf-tls)[TLS_AUTH]#own-certificate 2 pki:own-certificate/CA1
node(pf-tls)[TLS_AUTH]#own-certificate 3 pki:own-certificate/CA2
```

Configure the root CA's the Patton device shall trust. When the Patton device receives a certificate from a peer TLS endpoint, it only renders the certificate valid if it is signed by one of the listed root CAs:

```
node(pf-tls)[TLS_AUTH]#trusted-certificate pki:trusted-certificate/ROOT
```

Configure server authentication:

```
node(pf-tls)[TLS_AUTH]#no authentication incoming
node(pf-tls)[TLS_AUTH]#authentication outgoing
```

Trinity: Configure SIP with TLS

As a final step in the TLS configuration procedure on the Patton device you have to use the created TLS profile in one of the SIP gateways and enable TLS.

First, configure the TLS profile and transport on your SIP gateway. UDP and TCP transport may be disabled if you only want to use TLS. For example to (1) create a new SIP gateway GW_SIP, (2) use the TLS profile TLS_AUTH, (3) enable TLS as transport protocol only, and (4) let the gateway listen on the default TLS port 5061 on the IP address of the IP interface WAN, enter the following commands:

```
node(cfg)#context sip-gateway GW_SIP
node(sip-gw)[GW_SIP]#use profile tls TLS_AUTH
node(sip-gw)[GW_SIP]#interface GW_SIP_IF
node(sip-if)[GW_SIP.GW_SIP_IF]#transport-protocol tls
node(sip-if)[GW_SIP.GW_SIP_IF]#no transport-protocol udp+tcp
node(sip-if)[GW_SIP.GW_SIP_IF]#bind ipaddress ROUTER WAN WAN
```

Create a new SIP interface in context cs, set the Lync server host name as remote address and bind the interface to the created SIP gateway. Note that the remote name must be one of the DNS or Subject Alternate Names configured in the Lync server certificate.

```
node(cfg)#context cs SWITCH
node(ctx-cs)[SWITCH]#interface sip IF_SIP_TLS
node(if-sip)[SWITCH.IF_SIP_TLS]#remote sip.new.lync2013.com tls-port 7777
node(if-sip)[SWITCH.IF_SIP_TLS]#bind context sip-gateway GW_SIP
node(if-sip)[SWITCH.IF_SIP_TLS]#route call dest-interface ISDN
```

```

Enable SRTP in the VoIP profile.
node(cfg)#profile voip DEFAULT
node(pf-voip)[DEFAULT]#rtp security forced

```

Final Configuration on Trinity device

The following pages contain a complete startup-config file for a SN4980/4E120VR device. It contains all configuration commands required to connect the gateway to a Lync server. Note that the private key and the certificates do not appear in this configuration file. They are stored in persistent memory next to the startup-config file.

```

#-----#
# #
# Patton Electronics Company #
# SN4980/4E120VR #
# Generated configuration file #
# #
#-----#
cli version 4.00
  profile tls DEFAULT
  authentication incoming
  authentication outgoing
  private-key pki:private-key/DEFAULT
  own-certificate 1 pki:own-certificate/DEFAULT

profile tls TO_LYNC
  authentication incoming
  authentication outgoing
  private-key pki:private-key/SECRET1
  own-certificate 1 pki:own-certificate/CERT1
  trusted-certificate pki:trusted-certificate/ROOT

ntp
  server swisstime.ethz.ch
  no shutdown

profile voip DEFAULT
  codec 1 g711alaw64k rx-length 20 tx-length 20
  codec 2 g711ulaw64k rx-length 20 tx-length 20
  rtp security preferred

dns-server
  name-server address 172.16.75.202
  no shutdown

context ip ROUTER

  interface LAN
    ipaddress LAN2 172.16.44.111/19
    ipaddress DHCP

  routing-table DEFAULT

context cs SWITCH
  no shutdown

```

```
interface isdn IF_PRI0
  route call dest-interface IF_SIP_TLS
  user-side-ringback-tone
  inband-info accept force call-release

interface sip IF_SIP_TLS
  bind context sip-gateway GW_SIP_TLS
  route call dest-interface IF_PRI0
  remote patton.new.lync2013.com tls-port 7777
  hold-method direction-attribute sendonly
  prack accept required
  prack emit supported

context sip-gateway GW_SIP_TLS

  interface IF_GW_SIP_TLS
    no transport-protocol udp+tcp
    transport-protocol tls 5067
    bind ipaddress ROUTER LAN LAN2

context sip-gateway GW_SIP_TLS
  no shutdown

port ethernet 0 0
  bind interface ROUTER LAN
  no shutdown

port ethernet 0 1
  shutdown

port elt1 0 0
  port-type e1
  clock slave
  framing crc4
  encapsulation q921

q921
  uni-side auto

  q931
    protocol dss1
    uni-side user
    bind interface SWITCH IF_PRI0

port elt1 0 0
  no shutdown

port elt1 0 1
  port-type e1
  clock master
  framing crc4
  shutdown

port elt1 0 2
```

```
port-type e1
clock master
framing crc4
shutdown

port elt1 0 3
port-type e1
clock master
framing crc4
shutdown

configure
clock local default-offset +02:00
```


Chapter 56 **Secure SIP Applications**

Chapter contents

Introduction	726
TLS/SRTP encryption without TLS authentication	726
Configuration without TLS/SRTP	726
Tasks to Configure TLS/SRTP	728
Configure URI-scheme	728
Enable TLS on the SIP gateway	728
Enable SRTP on the VoIP profile	729
Configuration with TLS/SRTP	729
TLS/SRTP encryption with mutual TLS authentication (MTLS) based on exchanged, locally-generated, self-signed certificates	730
Tasks to create and exchange self-signed certificates.....	730
Generate private key	731
Generate certificate request	731
Self-sign the certificate request	731
Exchange certificates	731
Configure the TLS profile	731
Configuration with TLS/SRTP and mutual TLS authentication	732
TLS/SRTP encryption with mutual TLS authentication (MTLS) in a Microsoft Lync environment	733

Introduction

With Transport Layer Security (TLS) and Secure Real-Time Protocol (SRTP) Trinity devices are capable of securing both signaling connections and voice streams of SIP calls. Both protocols have to operate in harmony and work in a couple of different scenarios. Trinity, the software running on Patton devices offers a vast number of configuration options, depending on whether you just want to use TLS/SRTP to encrypt your calls or whether you would like to set up a proper trust relationship with certificates to be able to authenticate the SIP peer.

This chapter discusses different scenarios and provides the configuration snippets to set up the Patton device for them. In particular, this chapter covers the following scenarios:

- TLS/SRTP encryption without TLS authentication
- TLS/SRTP encryption with mutual TLS authentication (MTLS) based on mutually exchanged, locally-generated, self-signed certificates
- TLS/SRTP encryption with mutual TLS authentication (MTLS) in a Microsoft Lync environment.

TLS/SRTP encryption without TLS authentication

TLS connections always perform two tasks: symmetric packet encryption and endpoint authentication through certificates. You cannot have one without the other. Setting up a trust relationship with certificates is a complex task, described in the next scenarios. This scenario covers the case where you just want to encrypt SIP signaling traffic (and secure the RTP streams) without caring about TLS endpoint authentication. In this case it is sufficient to use a self-signed certificate. In a web-of-trust certificate scheme there is no central CA, and so identity certificates for each user can be self-signed. As the name tells it is an identity certificate that is signed by the same entity whose identity it certifies.

Patton devices come with an automatically and randomly generated private key and with a self signed certificate (with a modulus length of 512 bits). If you don't change the default TLS profile, this self-signed certificate will be used automatically for all TLS connections. Note that in this scenario, peer SIP devices will not be able to authenticate the Patton device over TLS, because the self-signed certificate is only stored locally and no trust relationship between the Patton device and the peer device exists. However, TLS still encrypts the signaling traffic, which may be sufficient in some basic scenarios.

Configuration without TLS/SRTP

In the following, you will learn how you can secure the SIP and RTP communication starting from an existing configuration that does not use TLS or SRTP yet. Figure 1 below depicts a LAN segment where two Patton SIP-to-SIP gateways are connected back-to-back. They are to secure a SIP connection between two SIP soft-phones that are not capable of doing TLS/SRTP.

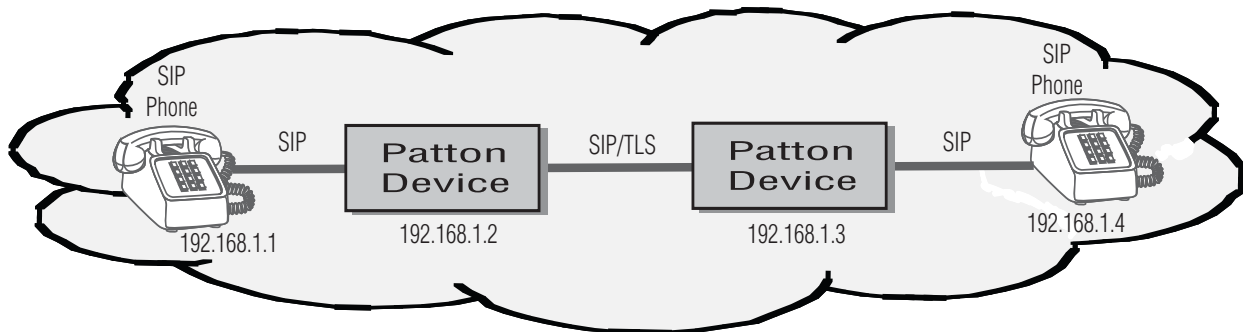


Figure 108. Patton device secures a SIP connection between two SIP soft-phones.

The devices encapsulate the plain SIP messages into a TLS connection and at the same time secure the voice streams using SRTP.

Let us assume that the two gateways are already configured to communicate via a non-secure SIP and RTP connection. Please find the configuration of the Patton device 192.168.1.2 before enabling TLS/SRTP below. Note that on first boot up the Patton device automatically created the *DEFAULT* private key, the *DEFAULT* own-certificate, and the *DEFAULT* TLS profile. Refer to the section “[Generated default files](#)” in Chapter 35, “[Public-Key Infrastructure \(PKI\)](#)” on page 408 for more information.

```

profile tls DEFAULT
  no authentication incoming
  no authentication outgoing
  private-key pki:private-key/DEFAULT
  own-certificate 1 pki:certificate/DEFAULT

profile voip DEFAULT
  codec 1 g711alaw64k rx-length 20 tx-length 20
  codec 2 g711ulaw64k rx-length 20 tx-length 20

context ip ROUTER
  interface LAN
    ipaddress LAN 192.168.1.2/24

context cs SWITCH
  no shutdown

interface sip IF_SIP_PHONE
  bind context sip-gateway GW_SIP
  route call dest-interface IF_SIP_TLS
  remote 192.168.1.1
  no call-transfer accept
  no call-transfer emit

interface sip IF_SIP_TLS
  bind context sip-gateway GW_SIP
  route call dest-interface IF_SIP_PHONE
  remote 192.168.1.3
  no call-transfer accept
  no call-transfer emit

```

```

context sip-gateway GW_SIP
  use profile tls DEFAULT

  interface GW_SIP_IF
    transport-protocol udp+tcp 5060
    no transport-protocol tls
    bind ipaddress ROUTER LAN LAN

context sip-gateway GW_SIP
  no shutdown

port ethernet 0 0
  bind interface ROUTER LAN
  no shutdown

```

Tasks to Configure TLS/SRTP

The following steps list all commands you will have to enter to configure the Patton device 192.168.1.2 in order to secure the connection to the peer device 192.168.1.3 by using TLS for signaling and SRTP for voice-data traffic:

- Configure URI-scheme (see page 728)
- Enable TLS on the SIP gateway (see page 728)
- Enable SRTP on the VoIP profile (see page 729)
- Configuration with TLS/SRTP (see page 729)

Configure URI-scheme

First, use the *sips* URI-scheme towards the peer Patton device. This makes sure that outbound requests are sent with a *sips* URI, which forces the gateway to use a secure connection such as TLS:

```

node>enable
node#configure
node(cfg)#context cs SWITCH
node(ctx-cs)[SWITCH]#interface sip IF_SIP_TLS
node(if-sip)[SWITCH.IF_SIP_TLS]#uri-scheme sips

```

Make sure that the URI scheme towards the connected SIP soft-phone is still set to *sip*, because we don't want to secure the connection to the soft-phone:

```

node(cfg)#context cs SWITCH
node(ctx-cs)[SWITCH]#interface sip IF_SIP_PHONE
node(if-sip)[SWITCH.IF_SIP_TLS]#uri-scheme sip

```

Enable TLS on the SIP gateway

Next, configure the transport protocol TLS on the interface of the SIP gateway.

```

node(cfg)#context sip-gateway GW_SIP
node(sip-gw)[GW_SIP]#interface GW_SIP_IF
node(sip-if)[GW_SIP.GW_SIP_IF]#transport-protocol tls

```

Enable SRTP on the VoIP profile

Finally, secure the stream of voice-data packets by configuring SRTP in the VoIP profile.

```
node(cfg)#profile voip DEFAULT
node(pf-voip)[DEFAULT]#rtp security preferred
```

The setting *preferred* means that we only use SRTP if the peer device supports it as well. This allows us to use the same VoIP profile for the *IF_SIP_TLS* as well as for the *IF_SIP_PHONE* interface, assuming that the soft phone just ignores the *crypto* attribute in the SDP part of the SIP message. If your soft-phone does not support the *crypto* attribute at all, you will have to create a separate VoIP profile and use in on the interface *IF_SIP_PHONE*.

Note You have to perform the same tasks on the other Patton device 192.168.1.3.

Configuration with TLS/SRTP

The text below includes all TLS/SRTP configuration tasks (highlighted in bold).

```
profile tls DEFAULT
  no authentication incoming
  no authentication outgoing
  private-key pki:private-key/DEFAULT
  own-certificate 1 pki:own-certificate/DEFAULT

profile voip DEFAULT
  codec 1 g711alaw64k rx-length 20 tx-length 20
  codec 2 g711ulaw64k rx-length 20 tx-length 20
  rtp security preferred

context ip ROUTER
  interface LAN
  ipaddress LAN 192.168.1.2/24

context cs SWITCH
  no shutdown

interface sip IF_SIP_PHONE
  bind context sip-gateway GW_SIP
  route call dest-interface IF_SIP_TLS
  remote 192.168.1.1
  no call-transfer accept
  no call-transfer emit
  uri-scheme sip

interface sip IF_SIP_TLS
  bind context sip-gateway GW_SIP
  route call dest-interface IF_SIP_PHONE
  remote 192.168.1.3
  no call-transfer accept
  no call-transfer emit
  uri-scheme sips

context sip-gateway GW_SIP
  use profile tls DEFAULT
```

```

interface GW_SIP_IF
  transport-protocol udp+tcp 5060
  transport-protocol tls 5061
  bind ipaddress ROUTER LAN LAN

context sip-gateway GW_SIP
  no shutdown

port ethernet 0 0
  bind interface ROUTER LAN
  no shutdown

```

TLS/SRTP encryption with mutual TLS authentication (MTLS) based on exchanged, locally-generated, self-signed certificates

This scenario extends the previous one by mutual TLS authentication. Instead of using certificates that have been signed by a proper Certification Authority (CA), we still use self-signed certificates here. For this purpose, we let the two Patton devices exchange their self-signed, locally-generated certificates via a TFTP server (see Figure 2). This is helpful if you want to avoid the complexity of a proper certification chain but still want to make sure that the two Patton devices authenticate themselves over TLS

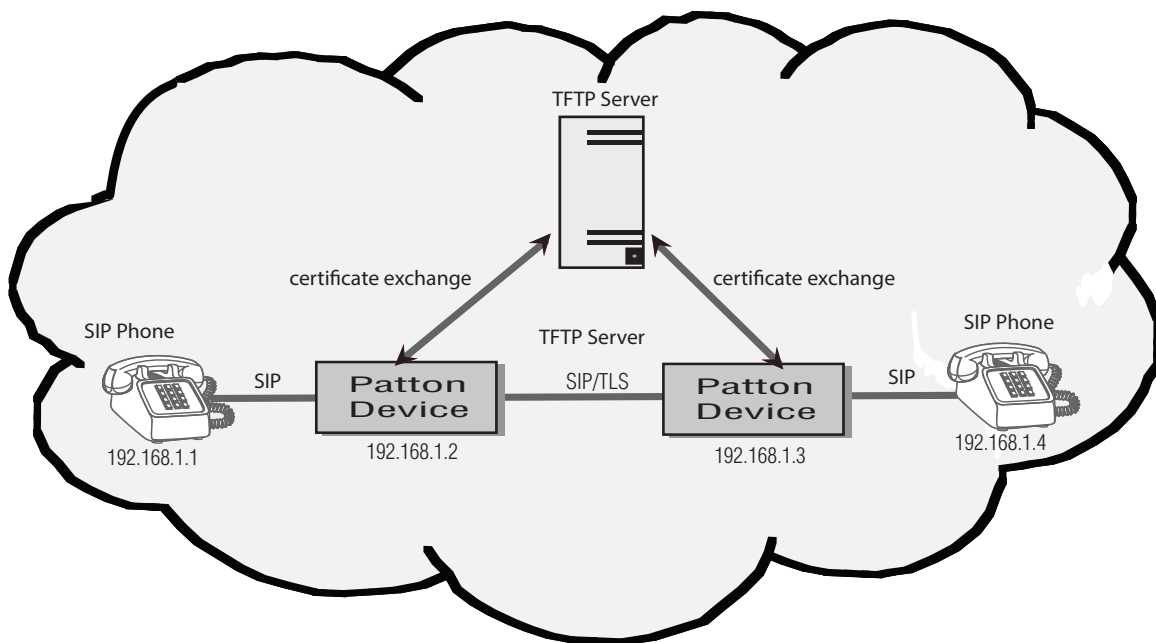


Figure 109. Patton devices secures a SIP connection between two SIP soft-phones.

The Patton devices encapsulate the plain SIP messages into a TLS connection and at the same time secure the voice streams using SRTP. The locally-generated, self-signed certificates are exchanged via a TFTP server prior to making calls.

Tasks to create and exchange self-signed certificates

The following steps start with the TLS configuration completed in the previous section.

Generate private key

We continue to configure the Patton device 192.168.1.1 and first create a new private key with longer modules:

```
node(cfg)#generate pki:private-key/SECRET1 key-length 2048
Generating RSA private key, 2048 bit long modulus
..+++
.....+++
e is 65537 (0x10001)
writing RSA key
```

Generate certificate request

As a next step, generate a certificate request derived from the created private key. The certificate request contains the same information as in the final certificate, but the request is not signed yet.

Note Note: The common-name at the end of the command must match the IP address of the Patton device.

```
node(cfg)#generate pki:certificate-request/REQUEST1 private-key
pki:private-key/SECRET1 country CH state Bern locality Bern organization
Patton organization-unit RND common-name 192.168.1.2
```

Self-sign the certificate request

You can now sign the certificate request locally with the same private key that we used to create it (self-signed). We set the validity period to one year (356 days). Note: the printed CN field must match the IP address of the Patton device:

```
generate pki:certificate/CERT1 certificate-request pki:certificate-request/
REQUEST1 private-key pki:private-key/SECRET1 validity-period 356
Signature ok
subject=/C=CH/ST=Bern/L=Bern/O=Patton/OU=RND/CN=192.168.1.2
Getting Private key
```

Exchange certificates

The next step is to deploy the certificate to the peer Patton device. Export the created owncertificate to your TFTP server (e.g. at 192.168.1.10) and import it on the peer device as trusted certificate:

```
node(cfg)#copy tftp://192.168.1.10/CERT1_FROM_192.168.1.3 pki:trusted-certificate/
CERT1_FROM_192.168.1.3
```

If all the steps above were executed for the device 192.168.1.3 as well, you now are able to import the remote certificate (stored as CERT1_FROM_192.168.1.3) from the TFTP server as trusted certificate:

```
node(cfg)#copy tftp://192.168.1.10/CERT1_FROM_192.168.1.3 pki:trustedcertificate/
CERT1_FROM_192.168.1.3
```

Configure the TLS profile

Create a new TLS profile and configure it with the new private key and the own-certificate of the device:

```
node(cfg)#profile tls TLS_AUTH
node(pf-tls)[TLS_AUTH]#private-key pki:private-key/SECRET1
```

```
node(pf-tls)[TLS_AUTH]#own-certificate 1 pki:certificate/CERT1
```

Configure the imported trusted certificate in the TLS profile.

```
node(pf-tls)[TLS_AUTH]#trusted-certificate pki:trustedcertificate/
CERT1_FROM_192.168.1.3
```

Enable mutual TLS authentication:

```
node(pf-tls)[TLS_AUTH]#authentication incoming
node(pf-tls)[TLS_AUTH]#authentication outgoing
```

Finally, use the new TLS profile on the sip-gateway.

```
node(cfg)#context sip-gateway GW_SIP
node(sip-gw)[GW_SIP]#use profile tls TLS_AUTH
```

Configuration with TLS/SRTP and mutual TLS authentication

The text below includes all configuration task discussed in this section (highlighted).

```
profile tls DEFAULT
  no authentication incoming
  no authentication outgoing
  private-key pki:private-key/DEFAULT
  own-certificate 1 pki:certificate/DEFAULT

profile tls TLS_AUTH
  authentication incoming
  authentication outgoing
  private-key pki:private-key/SECRET1
  own-certificate 1 pki:certificate/CERT1
  trusted-certificate pki:trusted-certificate/CERT1_FROM_192.168.1.3

profile voip DEFAULT
  codec 1 g711alaw64k rx-length 20 tx-length 20
  codec 2 g711ulaw64k rx-length 20 tx-length 20
  rtp security preferred

context ip ROUTER
  interface LAN
    ipaddress LAN 192.168.1.2/24

context cs SWITCH
  no shutdown

  interface sip IF_SIP_PHONE
    bind context sip-gateway GW_SIP
    route call dest-interface IF_SIP_TLS
    remote 192.168.1.1
    no call-transfer accept
    no call-transfer emit
    uri-scheme sip

  interface sip IF_SIP_TLS
    bind context sip-gateway GW_SIP
    route call dest-interface IF_SIP_PHONE
```



```
remote 192.168.1.3
no call-transfer accept
no call-transfer emit
uri-scheme sips

context sip-gateway GW_SIP
use profile tls TLS_AUTH

interface GW_SIP_IF
transport-protocol udp+tcp 5060
transport-protocol tls 5061
bind ipaddress ROUTER LAN WAN

context sip-gateway GW_SIP
no shutdown

port ethernet 0 0
bind interface ROUTER WAN
no shutdown
```

TLS/SRTP encryption with mutual TLS authentication (MTLS) in a Microsoft Lync environment

This scenario makes use of proper certificates, signed by an official Certification Authority (CA). The goal is to connect a Patton device as an Enterprise Session Boarder Router to a Microsoft Lync network. For this purpose, please read Chapter 56, "[Secure SIP Applications](#)" on page 725. It contains a detailed description of how to setup both the Lync server as well as the Patton device, including certificate handling.

Chapter 57 **Secure Real-Time Protocol (SRTP) Configuration**

Chapter contents

Introduction	735
About SRTP	735
SRTP configuration task list	735
License	735
Information about using DSP channels	735
Configure secure call signaling	736
Configure RTP security	736
Protection against key compromising through forking	738
Protection against key compromising through call forward	738

Introduction

This chapter provides an overview of Trinity's SRTP capabilities and describes the tasks involved in configuring it.

About SRTP

SRTP provides security, confidentiality, message authentication and replay protection for RTP and RTCP packets. This is achieved by generating a master key for each RTP stream, which is used to encrypt and decrypt the RTP packets. The keys are exchanged in the SDP content of the SIP signaling messages.

To protect the integrity of the SRTP master keys, securing the SIP signaling traffic is required too. This can be achieved by using TLS. Although not recommended it is still possible to use SRTP without TLS.

SRTP configuration task list

To configure the SRTP, perform the tasks in the following sections.

- License
- Information about using DSP channels
- Configure secure call signaling
- Configure RTP security
- Configure optional protection against key compromising through forking
- Configure optional protection against key compromising through call forward

License

The usage of a Patton device for doing SRTP, URI scheme sips and TLS are licensed. It is a precondition to have the license "sip-tls-srtp" installed on the Patton device before being able to perform all configuration steps.

Information about using DSP channels

A call encrypting its media-stream with SRTP always needs a DSP channel to perform the encryption of the outgoing SRTP stream and the decryption of the incoming SRTP stream. Therefore SRTP cannot be used on devices without DSP. The total number of DSP channels needed for one call depends on the involved protocols.

- For calls SIP to ISDN, SIP to FXS, SIP to FXO and vice versa there is always **one DSP channel** needed to do SRTP. That is the same number as when doing RTP without encryption.
- For calls SIP to SIP there are always **two DSP channels** needed to do SRTP. That is the same number as when doing transcoding from one codec to another codec.

When doing SRTP there is not the same possibility to save DSP channels in a SIP to SIP call as in a call without SRTP. The Patton device terminates always the SRTP streams.

Following an overview of using DSP channels in SIP to SIP calls:

RTP Security on SIP session toward A	RTP Security on SIP session toward B	Same or different codes on sessions	Used SDSP Channels
Unsecure RTP	Unsecured RTP	same	0
Unsecure RTP	Unsecure RTP	different	2
Unsecure RTP	SRTP	same	2
Unsecure RTP	SRTP	different	2
SRTP	SRTP	same	2
SRTP	SRTP	different	2

Configure secure call signaling

The overall security of the SRTP master keys depends on having the sip signaling secured. Therefore, using the sips URI scheme is encouraged because it does force the sip signaling to be secured with TLS until reaching the destination domain. Using the sips URI scheme requires having TLS transport configured for SIP. TLS can be configured and used without URI scheme sips. But consider: TLS secures signaling only until the next hop and if the next hop is a proxy it could forward SIP signaling messages in an unsecured way. See Chapter 54, “[TLS Configuration for SIP](#)” on page 682 for details about TLS and URI scheme.

Some of the RTP security commands modes are depending on the security of the call signaling. A call is considered as secure signaled if and only if the following two conditions are met.

- The URI scheme of the request URI is equal to “sips”
- The transport used for signaling is TLS

The reasoning why it is not enough to have TLS as transport is twofold. First TLS guarantees secure signaling only until the next hop and in case of a proxy the SRTP keys could be forwarded in an insecure manner toward the device terminating the SRTP stream. The second reason is the sequence of SDP offer construction and the selection of the transport protocol for signaling. The construction of the SDP offer happens first, for which we need to know if we want to offer SRTP. At this time the used URI scheme is known, but the finally selected transport could potentially change in the case of the URI scheme sip. It could be that a message is meant to be sent over an unsecured TCP transport, fails to be sent over TCP, and is finally sent over the secure transport TLS as a fallback.

Configure RTP security

There are 5 different modes how to support or force usage of SRTP:

- Disabled
- Preferred
- Enforced
- Preferred only for secured call signaling
- Enforced only for secured call signaling

Disabled: We don't offer SRTP and force the remote device to do unprotected RTP. If we get an offer with only SRTP it is rejected with a "488 not acceptable here" message.

Mode: Profile voip

Step	Command	Purpose
1	node(pf-voip)[name]# no rtp security	Disables the usage of SRTP at all and enforces to do unprotected RTP. This is the Default setting.

Preferred: We offer both, SRTP and unprotected RTP in the same offering and let the remote device decide on what to use. SRTP is the first in the ordering to indicate our preference for it. Incoming we accept any offer we receive: Only with SRTP, Only with unprotected RTP or offers with both. In the last case we select SRTP in our answer to the remote device.

Mode: Profile voip

Step	Command	Purpose
1	node(pf-voip)[name]# rtp security preferred	Enables SRTP and unprotected RTP. If both are available SRTP is selected.

Enforced: We don't offer unprotected RTP and force the remote device to do SRTP. If we get an offer with only unprotected RTP it is rejected with a "488 not acceptable here" message.

Mode: Profile voip

Step	Command	Purpose
1	node(pf-voip)[name] # rtp security forced	Disables the usage of unprotected RTP at all and forces to do SRTP.

Preferred only for secured call signaling: The behavior depends on the security of the call signaling. Intention is to offer or accept SRTP only when the keys are protected through secure signaling. For secure calls the same rules apply as when the option Preferred would be selected. For non-secure signaled calls the same rules apply as when the option Disabled would be selected.

Mode: Profile voip

Step	Command	Purpose
1	node(pf-voip)[name] # rtp security preferred sips	Enabled the usage of SRTP only for secured calls. Do unprotected RTP for non-secure calls.

Enforced only for secured call signaling: The behavior depends on the security of the call signaling. Intention is to enforce secure media when the keys are protected through secure signaling. For secure calls the same rules apply as when the option Enforced would be selected. For non-secure signaled calls the same rules apply as when the option Preferred would be selected.

Mode: Profile voip

Step	Command	Purpose
1	node(pf-voip)[name] # rtp security forced sips	Enforce the usage of SRTP only for secured calls. Do offer and accept unprotected RTP and SRTP for non-secure calls.

Protection against key compromising through forking

In a forking scenario there is the possibility of a key being compromised. We are party A sending an INVITE containing the SRTP master key A to a proxy. The proxy forks the INVITE to party B and C and thus B and C getting to know key A, which is used to encrypt the SRTP stream toward party A. Now B picks up the call and C is dropped off the call. This results in a call where C knows the key A and can listen to the SRTP stream from B toward A.

To avoid this we as party A can create a new key A2 and send it upon connecting with a re-INVITE message to party B. This guarantees that as soon as both parties are connected and talking C cannot listen anymore to B's speech. The disadvantage is that because we as party A have no possibility to detect all forking cases, we need to generate a new key and re-INVITE upon connecting always. Therefore this behavior can be enabled or disabled.

Mode: Interface sip

Step	Command	Purpose
1	node(if-sip)[name] # renew-srtp-keys-on-connect	Configures to always generate new SRTP master keys upon connect and sending with a re-INVITE. Default is disabled.

Protection against key compromising through call forward

In a call forward scenario there is the possibility of a key being compromised. We are party A sending an INVITE containing the SRTP master key A to party B. Party B forwards the call toward party C by sending a "302 temporarily moved" answer toward us. We send in some cases now the same INVITE request with the same key A to party C. This results in a call where B knows the key A and can listen to the SRTP stream from C toward A. In other cases we generate a new INVITE request with a new key A2 to be sent to party C, which results in a call where party B does not know any key.

The behavior of generating a new key or sending the same key twice depends on the configuration of call reroute and call redirect. We send the same key only if the commands "new-session-after-redirect" and "call-reroute accept" are both disabled, which is the default setting. To ensure the generation of a new key enable either "new-session-after-redirect", "call-reroute accept" or both. Be aware that these commands existed already before and have other effects apart from influencing SRTP key generation. If you are in doubt enable only "new-session-after-redirect".

Mode: Interface sip

Step	Command	Purpose
1	node(if-sip)[name] # new-session-after-redirect	Configures to always generate a new INVITE request from scratch, when being forwarded. With the effect that a new SRTP key is generated too. Default is disabled.

Mode: Interface sip

Step	Command	Purpose
1	node(if-sip)[name] # call-reroute-accept	Configures to forward the call toward call-control, when being forwarded. The call could be pushed back to another port or protocol. If the forwarded call is going to SIP again all rules the sip interface which is reached are applied. This could mean no SRTP is done at all. But if SRTP is offered again it is guaranteed to generate a new SRTP master key. Default is disabled.

Chapter 58 SIP Security

Chapter contents

Introduction	742
Transport Layer Security (TLS)	742
TLS configuration task list	742
Install license	742
Configure URI scheme	743
URI Scheme for Calls	743
SIP	743
SIPS	743
Transparent	743
URI Scheme for Registration	744
SIP	744
SIPS	744
URI Scheme for Re-routed Calls	744
Rerouted	744
Original	744
URI Scheme for Transferred Calls	745
Configure Time	745
Configure Public-Key Infrastructure (PKI)	745
About TLS Profiles	745
Basic Operation Principle of TLS	746
TLS without authentication (default behavior)	746
TLS authentication with self-signed certificates	746
TLS Authentication with 3rd Party Signed Certificates	747
TLS Profile Default	747
Configure TLS Profile	747
Creating a TLS profile and enter configuration mode	747
Private key	748
Own-certificate chain	748
Trusted Certificates	749
Authentication	750
Cipher Suites	751
Compression	752
Protocol	752
Configure TLS profile usage	752
Configure TLS transport	752
Configure transport enforcement or fallback	753
Configure Non-Default TLS Port Usage	755
Configure SRTP	756
Security consideration: Why you should use mutual TLS authentication	757

IP does not Re-use Connections for Requests in Different Directions	757
Two Scenarios of TLS Authentication Vulnerability	759
Conditions for the Patton device to Re-use Connections	759
Caveat for TLS Profile	760
Showing TLS profile	760
Troubleshooting	760
Check License	761
Check time	761
Check Certificates	761
Check TLS Profile Status	762
Check SIP Gateway Status	764
Check Server Name	764
Secure Real-Time Protocol (SRTP) Configuration	766
SRTP configuration task list	766
License	766
Information about using DSP channels	766
Configure secure call signaling	767
Configure RTP security	767
Protection against key compromising through forking	769
Protection against key compromising through call forward	769

Introduction

This chapter provides overviews and describes the tasks involved in configuring the following:

- Trinity’s Transport Layer Security (TLS) capabilities for SIP (see section “[Transport Layer Security \(TLS\)](#)”).
- Trinity’s Secure Real-Time Protocol (SRTP) capabilities (see section “[Secure Real-Time Protocol \(SRTP\) Configuration](#)” on page 766).

Transport Layer Security (TLS)

TLS provides communication security over insecure networks such as the Internet. TLS allows transport connections such as TCP to be safe from eavesdropping and tampering. For this purpose, TLS uses a Public Key Infrastructure (PKI) for authentication and confidentiality of the key exchange. Payload confidentiality is ensured with symmetric encryption using the exchanged keys. Message authentication and integrity is achieved by adding authentication codes to each message.

Using TLS for SIP signaling ensures that the content of the SIP messages is not subject to eavesdropping. This is especially important for SRTP master keys, which are exchanged over SIP/SDP signaling messages. TLS guarantees that the next hop destination is authenticated and that the messages are not altered by a third party.

TLS configuration task list

To configure TLS, perform the tasks in the following sections:

- Install license
- Configure URI scheme
- Configure Time
- Configure Public-Key Infrastructure (PKI)
- About TLS profiles
- Configure TLS profile
- Configure TLS-profile usage
- Configure TLS transport
- Configure transport-protocol enforcement or fallback
- Configure non-default TLS port usage
- Configure SRTP
- About security issue: connection re-use
- Troubleshooting

Install license

Voice security is a licensed option for Patton devices. You need to have a “sip-tls-srtp” license installed on your device before you are able to configure and use SRTP and TLS.

Configure URI scheme

The next step for enabling SIP over TLS is to configure the URI scheme for all SIP services that require a secure transmission. The URI scheme for SIP calls is to be configured on the SIP interface in context CS, whereas the URI scheme for SIP registrations is configured on the outbound registration mode in the context SIP gateway. This separation gives you the flexibility to decide individually whether or not call or registration connections will be encrypted.

URI Scheme for Calls

For calls there are three possible settings for the URI scheme.

- SIP
- SIPS
- transparent

SIP. The SIP URI scheme defines an unsecure SIP URI, i.e., “sip:200@patton.com”. Calls destined to a request-URI with the SIP URI scheme usually are signaled over non-secure transports as UDP and TCP. However, they may be signaled over secure transports such as TLS as well. The transports depend on the availability and preference of transport protocols for reaching the given URI. For example, a DNS lookup for an URI according to the SIP URI scheme may return a list of transport protocols, including TLS. Our protocol preference selection configurable in the call-outbound configuration mode of the location service – then defines which of the transports is preferred.

SIPS. The SIPS URI scheme defines a secure SIP URI, i.e., “sips:200@patton.com”. Calls destined to a request-URI with the SIPS URI scheme must be signaled over the secure transports TLS. (Although there are alternative secure transports protocols, Patton devices only support TLS at the moment).

Transparent. Selects the URI scheme on a per-call basis. The actual URI scheme for an outgoing call is defined by the call-control peer, i.e., over the protocol where the call entered the Patton device. For all protocols other than SIP, when there isn't any security information – we always use the SIP URI scheme. However, when an incoming SIP call is routed over SIP again, the URI scheme is tunneled transparently through our call-control application. If the requested URI of the incoming call is SIPS and the transport protocol is TLS, then the outgoing call will also be placed using the SIPS URI scheme. However, when the incoming request-URI is SIP or when it arrives over a non-secure transport the resulting URI scheme is SIP.

Mode: Interface SIP

Step	Command	Purpose
1	<code>nodeif-sip)[name]# uri-scheme { sip sips transparent }</code>	Configures the URI scheme for outgoing calls. Default is transparent.

Note If you are using SRTP it is strongly recommended to signal calls over a secure TLS connection by using the SIPS URI scheme. This is because the SRTP session's master key is exchanged in the SDP portion of SIP signaling messages. If an insecure transport is used for SIP signaling, an eavesdropper may read the SRTP master key and listen to the exchanged RTP voice packets. See “Configure SRTP” below for more information.

URI Scheme for Registration

The URI scheme for outbound registrations can be configured separately from the URI scheme for outbound calls.

Note For registration, the transparent option is not available, because there isn't any incoming source for registrations.

- SIP
- SIPS

SIP. The SIP URI scheme defines an insecure SIP URI, for example “sip:200@patton.com”. The registration messages and incoming calls to the registered contact *may* travel over any secure or non-secure transport.

SIPS. The SIPS URI scheme defines a secure SIP URI, for example “sips:200@patton.com”. The registration messages and incoming calls to the registered contact *must* use the secure transport protocol TLS.

Mode: Registration outbound

Step	Command	Purpose
1	<code>node(regout)# uri-scheme { sip sips }</code>	Configures the URI scheme for outbound registrations and the corresponding calls toward the registered contact. Default is sip.

URI Scheme for Re-routed Calls

It is possible to specify the behavior regarding URI scheme for rerouted calls. This is especially valuable when the call is rerouted to a URI with a different URI scheme than the original URI of the call.

- Rerouted
- Original

Rerouted. The URI scheme is taken from the reroute operation. This means it could be degradation or an enhancement of security of the call. Calls with a SIP URI could be degraded to a SIP URI or calls with a SIP URI could be changed to a SIP URI. This is the default behavior.

Original. The URI scheme for the rerouted call remains the same as for the original call. This means the rerouting operation cannot have an impact on the security of the call. Neither degradation or an enhancement. There is the possibility that the target destination is not compatible to the original URI scheme and would need the URI scheme from the reroute operation.

Mode: Interface SIP

Step	Command	Purpose
1	<code>node(if-sip)[name]# call-reroute uri-scheme { rerouted original }</code>	Configures the URI scheme for outgoing calls which are rerouted. Default is rerouted.

URI Scheme for Transferred Calls

For calls it is possible to specify the behavior regarding URI scheme in the case of a transfer of a call. This is especially valuable when the call is transferred to a URI with a different URI scheme than the original URI of the call.

- Referred
- Original

Referred The URI scheme is taken from the REFER request. This means it could be a degradation or an enhancement of security of the call. Calls with SIPs URI could be degraded to SIP URI or calls with SIP URI could be changed to SIPs URI. This is the default behavior.

Original The URI scheme for the transferred call remains the same as for the original call. This means the transfer operation cannot have an impact on the security of the call. Neither degradation or an enhancement. There is the possibility that the target destination is not compatible to the original URI scheme and would need the URI scheme from the transfer operation.

Mode: Interface SIP

Step	Command	Purpose
1	<code>node(if-sip)[name]# call-transfer uri-scheme { referred original }</code>	Configures the URI scheme for calls which are transferred. Default is referred.

Configure Time

The system time must be configured before you can work with certificates, and enabling the PKI and TLS. It is also very important to set the correct offset of the local time with respect to the UTC time, due to the validity of certificates expressed in UTC time. The configuration option that works best is to update the UTC time from a configured SNTP server and to derive the local time by configuring the static offset of the local time zone with respect to UTC. If a SNTP server is not available, you should first configure the offset of the local time zone and afterwards set the local time with the **et clock** command.

Note It is important to also correctly configure the time of remote devices, which are having TLS connections to our device.

Configure Public-Key Infrastructure (PKI)

The commands that are needed to set up the public-key infrastructure are described in detail in The PKI is responsible for creating, importing and exporting keys and certificates, and storing them persistently in flash memory.

A TLS profile (described in the next section) groups security and other relevant configuration required to open a TLS connection. For this purpose, the TLS profile may refer to a private, an own certificate and many trusted certificates. In order to configure and use a TLS profile, you need to create and/or import keys and certificates. Refer to “Public-Key Infrastructure (PKI)” on page 408 for more information.

About TLS Profiles

A TLS profile is a collection of TLS configuration parameters, which are used by a SIP gateway to establish a secure transport connection to a remote device. Therefore each SIP gateway is bound to one TLS profile. If

your device has to use different PKI keys or certificates to talk to different SIP servers, you have to configure multiple SIP gateway instances, each bound to its own TLS profile.

Basic Operation Principle of TLS

The server side of a TLS connection sends a certificate to the client that serves two purposes: encryption and authentication. First, the certificate contains a public key required to securely negotiate a symmetric session key to encrypt SIP signaling messages. Second, the certificate provides message authentication and integrity by identifying the server. The client is able to authenticate the server by verifying that the server's certificate is derived from one of the locally configured trusted certificates. At the same time, the client is able to verify that the TLS connection is actually terminated by the IP address or domain name listed in the *common name* field of the certificate. The server may request an optional certificate from the client, which is also called mutual authentication.

Encryption usually works with the DEFAULT TLS profile, i.e. without configuring any further options. However, for the authentication and integrity check to work, you have to configure the TLS profile to reflect the trust relationship with the servers/domains you want to communicate with, and you have to manually enable authentication for inbound and outbound connections.

For Trinity, you are able to establish different levels of security; you can set up a quick solution that works with almost zero configuration effort, or you can set up a secure trust relationship in the TLS profile. The following scenarios elaborate more on these security options.

TLS without authentication (default behavior)

This scenario requires no effort to install certificates and only a minimal effort to configure the device, but it also provides the lowest level of security. Each TLS-licensed Patton devices comes with a private key and a default self-signed certificate already installed. The **DEFAULT** TLS profile is configured to use this **DEFAULT** private key and **DEFAULT** own certificate to encrypt SIP signaling messages.

In terms of authentication and integrity check, the security provided by the unaltered TLS profile **DEFAULT** is pretty poor, because authentication for incoming and outgoing TLS connections is switched off. As a consequence, when we accept an incoming TLS connection, Trinity accepts any server certificate without checking a trust relationship and without checking whether the remote IP address matches the *common name* field in the server's certificate. If Trinity acts as a TLS server, the **DEFAULT** TLS profile does not force mutual authentication and thus does not check the identity of the connected client.

Thus, the unaltered TLS profile **DEFAULT** may be helpful for a quick start, but for a serious installation, you should configure a proper trust relationship as described below.

TLS authentication with self-signed certificates

In this scenario we increase the level of security provided by TLS compared to the previous zero-configuration scenario. This is achieved by creating a new self-signed certificate that is only valid for the actual IP address or domain name of the Patton device. See "[Public-Key Infrastructure \(PKI\)](#)" on page 408 to learn how to perform the steps described below:

1. Create a new certificate request, which contains as *common name* field with our public IP address or domain name.
2. Assign a **DEFAULT** private key to the request (or, optionally, assign a newly generated private key).
3. Configure the TLS profile to use the new self-signed certificate and the private key.

Note If such a TLS profile is bound to a SIP gateway, we send this self-signed certificate to identify ourselves. The remote TLS endpoints then are able to verify whether this device's IP address or domain name matches the sent certificate. In order for that to be secure, you also have to export the self-signed certificate and import it to all remote devices as a trusted certificate.

If the device authenticates remote devices, it needs to import the certificates of all remote devices and list them as *trusted certificates* in its TLS profile. In addition, it will switch-on incoming and outgoing authentication to verify whether the remote certificate is derived from one of the trusted certificates listed in the TLS profile. The outgoing authentication process verifies that the remote server endpoint's IP address or domain name matches the common name in the received certificate.

TLS Authentication with 3rd Party Signed Certificates

A generic certification chain consists of the certificate C itself, zero to many certificates of intermediate certification authorities CA1, CA2,..., CAn, and the certificate of the root certification authority CAroot.

In such a case, store certificate C and all certificates of the intermediate certification authorities CA1, CA2,..., CAn on the device that needs to be authenticated. The certificate of the root certification authority CAroot, on the other hand, must be stored as trusted certificate on the device that wants to authenticate the other device(s).

TLS Profile Default

When booting a TLS-licensed Patton device for the first time, a TLS profile with the name DEFAULT is created automatically; it is linked to an automatically-created private key DEFAULT and a self-signed certificate DEFAULT. It is configured not to authenticate and can be used to do authentication-free TLS. For enabling authentication and integrity checks, alter the DEFAULT profile or create a new TLS profile (recommended) and to enable inbound and/or outbound authentication. If a TLS profile is not explicitly bound to a SIP gateway, the DEFAULT TLS profile is used.

Configure TLS Profile

This section describes how to configure TLS profiles, which are those instances in your configuration that collect all TLS-related configuration parameters, and which are used by SIP gateways to establish secure transport connections to remote devices.

Creating a TLS profile and enter configuration mode

This procedure describes how to create a TLS profile and enter the TLS profile configuration mode.

Mode: Administrator exec

Step	Command	Purpose
1	node(cfg)#profile tls <i>name</i>	Creates the TLS profile <i>name</i> and enters its configuration mode.

The parameter *name* is the identifier by which the profile will be known. Entering this command puts you into the TLS profile configuration mode where you can enter the individual TLS configuration parameters.

Use the **no** form of this command to delete a TLS profile. You cannot delete a TLS profile if it is currently bound to a SIP gateway.

To modify a TLS profile that is currently being used by one or more active SIP gateways, the gateways are restarted gracefully; ongoing calls over a certain gateway still use the old TLS configuration until all calls over that gateway have terminated. At that point in time the gateway restarts and all future calls will use the new TLS configuration.

Example: Create a TLS profile

In the following example the TLS profile named *Y_TLS* has been created.

```
node>enable
node#configure
node(cfg)#profile tls MY_TLS
node(pf-tls)[MY_TLS]#
```

Private key

A private key file actually contains a private/public key pair. TLS requires them to secure its key exchange. Thus, it is important to link a TLS profile to a valid private key, which is done by the following command:

Mode: Profile TLS

Step	Command	Purpose
1	<code>node(pf-tls)[name]# [no] private-key pki:private-key/key</code>	Configures the private (and public) key to be used for the TLS key-exchange encryption. This configuration is mandatory!

Note If your TLS profile has a configured chain of *own-certificates* it is required that the first certificate in that chain has been created from a certification request, which contains the corresponding public key.

Example: Create a new private key and link it to a new TLS profile.

In the following example, we let Trinity generate a new private key. We then create a new TLS profile, called *Y_TLS* and link it to the generated private key.

```
node>enable
node#generate pki:private-key/MY_PRIVATE_KEY
node#configurenode(cfg)#profile tls MY_TLS
node(pf-tls)[MY_TLS]#private-key pki:private-key/MY_PRIVATE_KEY
```

Own-certificate chain

Our own certificate is used to identify our device to others when making or accepting TLS connections. The chain of own certificates consists of an order list of one or more links to stored certificates. The first certificate in the list (index 1) is used to communicate our public key to remote TLS devices. It is required that the private key linked by the TLS profile corresponds to the public key in this certificate. Therefore it is mandatory to configure a link to a private key on the TLS profile.

If you want to use a self-signed certificate, the chain of own certificates only contains one entry – a link to that certificate. Otherwise, if your certificate is signed by one or more levels of certification authorities, those intermediate certificates have to follow in the chain. That is, the next entry with index 2 links to the certificate of the certification authority that signed our own certificate at index 1. Then, the entry with index 3 links to the

certificate of the next higher-level certification authority, and so on. The certificate of the root certification authority is not required and may be omitted or may be configured as the final entry in the list.

Mode: Profile tls

Step	Command	Purpose
1	node(pf-tls)[name]# [no] own-certificate [index] pki: own-certificate /certificate	Configures the chain of own certificates. Inserts the certificate at the specified index into the chain or appends it at the end if no index is specified. The sequence of own certificates is of importance! This configuration is mandatory.

Example: Configure a chain of own certificates.

In the following example, we assume that the own-certificate as well as all intermediate certificates that signed our own certificate already have been imported (e.g. with the **own** command). We configure our TLS profile with the certificate chain that represents our own certificate.

```
node(cfg)#profile tls MY_TLS
node(pf-tls)[MY_TLS]#own-certificate 1 pki:certificate/MY_CERTn
ode(pf-tls)[MY_TLS]#own-certificate 2 pki:certificate/CERT_OF_MY_CA
node(pf-tls)[MY_TLS]#own-certificate 3 pki:certificate/CERT_OF_MY_CAS_CA
```

Trusted Certificates

A set of trusted certificates is required to express the trust relationship to other devices. When our device wants to authenticate other devices our TLS profile must be configured with the certificates of all corresponding trusted root certification authorities. In this case a set of one or more trusted certificates is needed. However, when our device doesn't need to authenticate other devices no trusted certificates are needed.

As a default behavior all imported trusted certificates are considered trusted when not configured otherwise in the TLS profile. That is, trusted root certificates usually only have to be imported (e.g. with the **own** command) but must not be linked explicitly in the TLS profile.

Note Certificates stored as own certificates are not automatically considered as trusted certificates. If you want to use an own certificate as a trusted certificate as well, you have to copy it to the trusted-certificate section.

Mode: Profile tls

Step	Command	Purpose
1	node(pf-tls)[name]# trusted-certificate all-stored	Configures to use all imported trusted certificates for the current TLS profile. This is the default setting.

In the case that you want to create different TLS profile where each of them links to a different set of trusted certificates, you have to configure those links explicitly.

Mode: Profile tls

Step	Command	Purpose
1	node(pf-tls)[name]# [no] trusted-certificate pki:trusted-certificate/certificate	Adds or removes a certificate to or from the set of trusted certificates of the current PKI profile. Removing the last link sets the TLS-profile configuration back to the default setting, which is to use all trusted certificates in PKI.

Example: Configures a set of trusted profiles.

In the following example, we want to limit the set of trusted devices to those that have their certificate derived from one of the following root certificates.

```
node(cfg)#profile tls MY_TLS
node(pf-tls)[MY_TLS]#trusted-certificate pki:trusted-certificate/ROOT_CA_COMPANY_A
node(pf-tls)[MY_TLS]#trusted-certificate pki:trusted-certificate/ROOT_CA_COMPANY_B
node(pf-tls)[MY_TLS]#trusted-certificate pki:trusted-certificate/ROOT_CA_OUR_OWN_-
COMPANY
```

Authentication

TLS-licensed Patton devices can be configured whether or not to authenticate other devices when establishing TLS connections. If TLS authentication is to be enabled, it is required to have at least one trusted certificate installed on the device and to link the TLS profile to that certificate accordingly (see the description of the **trusted-certificate** command above).

Note This device can only authenticate other devices, but it cannot force other devices to authenticate us. Therefore mutual TLS authentication depends on having each device configured to authenticate the other devices.

Although it is technically possible to authenticate only TLS connections where we are the server side or to authenticate only TLS connections where we are the client side it is not recommended to do so, because in a SIP call, the client and server role may change during a call. (For example, we are the client side for an outgoing call, but the server side if the call is terminated by the remote party.)

Furthermore, having a different configuration for incoming and outgoing TLS connections opens the security hole of abusing connection-reuse for getting or receiving calls to or from unauthenticated remote devices.

Mode: Profile TLS

Step	Command	Purpose
1	node(pf-tls)[name]# [no] authentication incoming	Configures whether or not we are authenticating the remote client in a new TLS connection. This means that we are acting as TLS server for that connection. If enabled, we verify that the received certificate is signed by one of the trusted certificates listed in this TLS profile. Default is enabled.

Step	Command	Purpose
2	node(pf-tls)[name][name]# [no] authentication outgoing	Configures whether or not we are authenticating the remote server in a new TLS connection. This means that we are acting as TLS client for that connection. If enabled, we verify: <ul style="list-style-type: none"> • The received certificate is signed by one of the trusted certificates listed in this TLS profile • The <i>common name</i> field of the received certificate matches the remote IP address or DNS name of the remote TLS endpoint Default is enabled.

Cipher Suites

TLS automatically negotiates to use the crypto algorithms that provide the strongest security, meaning that the remote endpoint may lower the level of security by only providing a weaker crypto algorithm. In high-security environments it may be desirable to restrict the set of offered crypto algorithms. This can be done with the **cipher-suites** command.

Instead of listing each algorithm individually, Trinity offers a selection of pre-configured cipher-suites. Either one or multiple suites can be selected. The suits themselves may overlap with other cipher-suites. The available suites are:

- **ALL** All cipher suites except the eNULL cipher. This is the default setting.
- **DEFAULT** A pre-selected group of ciphers that offer medium to high level of security and at the same time maximum interoperability in most scenarios.
- **HIGH** ciphers with a key-size of 128 bit or more.
- **MEDIUM** ciphers with a key-size of 128 bit.
- **LOW** ciphers with a key-size of 64 or 56 bit.
- **SHA1** ciphers using SHA-1 as digest algorithm.
- **MD5** ciphers using MD5 as digest algorithm.
- **aRSA** ciphers offering RSA authentication.
- **kRSA** ciphers offering RSA key exchange.
- **eNULL** ciphers offering no encryption. Not recommended, use only for debugging.

Mode: Profile tls

Step	Command	Purpose
1	node(pf-tls)[name]# cipher-suites (ALL { DEFAULT HIGH MEDIUM LOW SHA1 MD5 aRSA kRSA eNULL }*)	Configures one or multiple groups of cipher-suites to offer or accept. Default is ALL.

Compression

Compression has the benefit of reducing the size of the TLS packets and thus reducing the bandwidth needed for signaling over TLS. But it could have a negative effect on the performance for encrypting and decrypting the packet and therefore the load of SIP signaling traffic which can be handled on the device.

Mode: Profile TLS

Step	Command	Purpose
1	node(pf-tls)[name]# [no] compression	Configures if we are offering or accepting compression for TLS messages. Default is disabled.

Protocol

The Patton device supports SSL version 3 and its successor TLS version 1. The offering or acceptance can be configured separately for both of them. At least one of the Protocols need to be enabled for TLS operation.

Mode: Profile TLS

Step	Command	Purpose
1	node(pf-tls)[name]# [no] protocol tls-v1	Configures if we are offering or accepting the TLS protocol version 1. Default is enabled.
2	node(pf-tls)[name]# [no] protocol ssl-v3	Configures if we are offering or accepting the SSL protocol version 3. Default is enabled.

Configure TLS profile usage

To actually use TLS for SIP calls and/or registrations, a TLS profile must be bound to a SIP gateway. This allows to quickly changing between different TLS profiles. The DEFAULT profile may be bound if using the default private key and self-signed certificate is fine. Otherwise, if you want to configure a proper trust relationship or use different TLS profiles on different SIP gateways, you have to create new TLS profiles and use them on the respective SIP gateway with the **se** command:

Mode: Context sip-gateway

Step	Command	Purpose
1	node(sip-gw)[name]# use profile tls <name>	Configures which TLS profile to use for connections from and to this SIP gateway. Default is the TLS profile DEFAULT.

Configure TLS transport

Some TLS-related parameters have to be configured outside the TLS profile. In order for the TLS protocol to be used on a SIP gateway, you have to enable it and specify the TCP port over which TLS connections are accepted (port 5061 by default).

It is possible to configure the availability of UDP/TCP, and TLS separately on the transport interface of the SIP gateway. This for example allows a secure configuration where both UDP and TCP are disabled, whereas TLS is enabled as the only transport protocol.

Note In order to use the SIPs URI scheme, TLS must be enabled on the transport interface of the SIP gateway.

Mode: interface sip-gateway

Step	Command	Purpose
1	node(sip-if)[name]# [no] bind ipaddress [<context>] <interface> <label>	Binds the transport interface to a specific IP address on an IP interface in the selected context.
2	node(sip-if)[name]# [no] transport-protocol udp+tcp [<port>]	Enables the reception and sending of SIP requests over UDP and TCP from and to the specified port. Default is enabled. Default port is 5060.
3	node(sip-if)[name]# [no] transport-protocol tls [<port>]	Enables the reception and sending of SIP requests over TLS from and to the specified port. Default is disabled. Default port is 5061.

Configure transport enforcement or fallback

Having multiple transport protocols available to choose from when sending a SIP message makes it desirable to select the order in which they are being used, or even the possibility to force one specific transport protocol. The transport enforcement or fallback configuration only has an effect to outgoing requests to the next hop device. It is not possible to prefer a transport protocol for incoming requests, but TLS can be forced by disabling UDP and TCP on the transport level altogether. Before you are able to force a transport protocol you have to enable the protocol on the transport level (see above). Otherwise requests cannot be sent over the protocol at all.

Note You are able to configure the device to only use SRTP when its session keys can be exchanged over a secure signaling connection. Forcing TLS as transport protocol here is not enough for SRTP to recognize a secure connection. The only way to actually prove SRTP that it is using a secure connection is to configure the SIPs URI scheme. By setting the URI scheme to SIPs, we also force the transport protocol to be TLS, too, but in addition, the transport protocol for SIP signaling messages is forced to be secure beyond the next-hop device. The same applies for having TLS as the preferred transport protocol, where additionally a local fallback to an insecure transport protocol could happen.

The command for selecting a preferred transport protocol allows you to select as a second choice the first fallback and as a third choice the second fallback protocol. As there is a maximum of three protocols the second fallback is implicitly given by choosing the preferred and the first fallback.

Note Not specifying the fallback protocols doesn't disable them, but the order is derived from the internal default ordering. Specifying the transport protocols here does not enable them if they are disabled on the gateway's transport level. The preference of transport has no influence on calls with the URI scheme SIPs. Such calls are always forced to be signaled over TLS. The default ordering of preference is UDP, TCP, and TLS.

Mode: Call Outbound

Step	Command	Purpose
1	node(callout)# [no] transport-protocol force { udp tcp tls }	Forces outgoing INVITE requests to be sent over UDP, TCP, or TLS. If the according transport is not enabled on the sip-gateway level or the remote device does not support it there is no fallback to another transport.
2	node(callout)# [no] transport-protocol prefer { udp tcp tls } *	Sets the preferred transport protocol to choose if multiple are available for an INVITE request. Also specifies to which first and second alternative transport protocol we fall back. Default is to give preference to UDP, and to use TCP and then TLS as fallback.

The same protocol selection can be configured for outbound registrations as well by enforcing or preferring the transport protocol for REGISTER request. The same mechanism is applied for outbound registrations (REGISTER requests) as for outbound calls (INVITE requests).

In addition, we are able to control the transport protocol of incoming calls by the outbound registration; the registered contact address is stored on the server and remembers the transport protocol that was used to register the address. The server then uses this transport protocol to make calls, which the device will receive incoming calls for that registration with the transport protocol the registration was made with.

Mode: Registration outbound

Step	Command	Purpose
1	node(regout)# [no] transport-protocol force { udp tcp tls }	Forces outgoing REGISTER requests to be sent over UDP, TCP, or TLS. If the according transport is not enabled on the sip-gateway level or the remote device does not support it there is no fallback to another transport.
2	node(regout)# [no] transport-protocol prefer { udp tcp tls } *	Set the preferred transport protocol to choose if multiple are available for a REGISTER request. Also specifies to which first and second alternative transport protocol we fall back. Default is to give preference to UDP, and to use TCP and then TLS as fallback.

Configure Non-Default TLS Port Usage

Several SIP-related commands select the remote endpoint's host name and port number. All of them were extended for TLS and now allow you to optionally configure the remote TLS port next to the UDP/TCP port.

For most of these commands the port can be omitted because the standard SIP UDP and TCP port 5060 is used. For cases where the remote device does not use the standard port, the user must specify in the command. Now with the addition of TLS and the possibility to have a transport protocol fallback from TLS to UDP or TCP and vice versa, we need the ability to specify two ports: one for UDP and TCP and the other for TLS.

For all the following commands the same rule applies; the optional port parameter right after the ip-address parameter specifies the port number for UDP and TCP. If not indicated, the port number defaults to 5060. However, this first port parameter is never used for TLS, even if it is the only port specified explicitly, or if UDP and TCP transports are not available.

An additional parameter optionally configures the TLS port. If not indicated, the port defaults to 5061. This parameter is never used for UDP or TCP, even if it is the only port specified explicitly, or if the TLS transport is not be available.

Mode: Interface SIP

Step	Command	Purpose
1	node(if-sip)[name]# [no] remote <host> [<port>] [tls-port <tls-port>]	The existing remote command has the ability to specify a non-default UDP and TCP port and to specify a non-default TLS port. Default UDP/TCP port is 5060. Default TLS port is 5061.
2	node(if-sip)[name]# [no] address-translation outgoing to-header host-part fix <host> [<port>] [tls-port <tls-port>]	The existing address translation to-header command has the ability to specify a non-default UDP and TCP port and to specify a non-default TLS port. Default UDP/TCP port is 5060. Default TLS port is 5061.
3	node(if-sip)[name]# [no] address-translation outgoing request-uri host-part fix <host> [<port>] [tls-port <tls-port>]	The existing address translation Request-URI command has the ability to specify a non-default UDP and TCP port and to specify a non-default TLS port. Default UDP/TCP port is 5060. Default TLS port is 5061.

Mode: Service sip-conference

Step	Command	Purpose
1	node(svc-ls)[name]# [no] conference-server <host> [<port>] [tls-port <tls-port>]	The existing conference-server command has the ability to specify a non-default UDP and TCP port and to specify a non-default TLS port. Default UDP/TCP port is 5060. Default TLS port is 5061.

Mode: Call Outbound

Step	Command	Purpose
1	node(callout)# [no] force-destination <host> [<port>] [tls-port <tls-port>]	The existing force-destination command has the ability to specify a non-default UDP and TCP port and to specify a non-default TLS port. Default UDP/TCP port is 5060. Default TLS port is 5061.
2	node(callout)# [no] proxy <host> [<port>] [tls-port <tls-port>]	The existing proxy command has the ability to specify a non-default UDP and TCP port and to specify a non-default TLS port. Default UDP/TCP port is 5060. Default TLS port is 5061.

Mode: Registration Outbound

Step	Command	Purpose
1	node(regout)# [no] registrar <host> [<port>] [tls-port <tls-port>]	The existing registrar command has the ability to specify a non-default UDP and TCP port and to specify a non-default TLS port. Default UDP/TCP port is 5060. Default TLS port is 5061.
2	node(regout)# [no] force-destination <host> [<port>] [tls-port <tls-port>]	The existing force-destination command has the ability to specify a non-default UDP and TCP port and to specify a non-default TLS port. Default UDP/TCP port is 5060. Default TLS port is 5061.
3	node(regout)# [no] proxy <host> [<port>] [tls-port <tls-port>]	The existing proxy command has the ability to specify a non-default UDP and TCP port and to specify a non-default TLS port. Default UDP/TCP port is 5060. Default TLS port is 5061.

Mode: Message Inbound

Step	Command	Purpose
1	node(msgin)# [no] message-server <host> [<port>] [tls-port <tls-port>]	The existing message-server command has the ability to specify a non-default UDP and TCP port and to specify a non-default TLS port. Default UDP/TCP port is 5060. Default TLS port is 5061.

Configure SRTP

After having configured a secure URI scheme and a secure transport protocols for SIP signaling, you should consider securing the actual media-streams that carry voice or data by using SRTP. For more details, see [“Secure Real-Time Protocol \(SRTP\) Configuration”](#) on page 766.

Security consideration: Why you should use mutual TLS authentication

In the TLS profile, you are able to enable inbound and outbound TLS authentication separately (commands **authentication inbound** and **authentication outbound** respectively). This configuration affects how TLS connections are authenticated, i.e., whether the Patton device verifies TLS server certificates (for outgoing TLS connections) and/or whether it requests and verifies TLS client certificates (for incoming TLS connections). In this section, we will explain why you should enable TLS authentication in both directions for maximum security.

SIP gateway tries to re-use transport connections whenever possible; this is a desired and intended behavior according to the SIP standard. For example, if the Patton device opens ten calls to the exact same SIP server, it would be a waste of resources to establish a new TLS connection for each call. However, since each TLS connection requires CPU resources and network bandwidth to generate keys and negotiate security options, respectively.

The TLS authentication procedure, i.e. the verification of certificates, is only performed when the TLS connection is established. Since TLS connections can be re-used for SIP requests in both directions, not every single SIP call is authenticated. Furthermore, a TLS connection may be used to make calls (or other requests) in the opposite direction, which potentially cancels the intent of your authentication configuration.

IP does not Re-use Connections for Requests in Different Directions

A TLS connection may be opened in either direction during a call: one connection from the calling to the called party, and another one from the called to the calling party. Figure 1 depicts a common registration and call scenario without (left) and with (right) mutual TLS authentication.

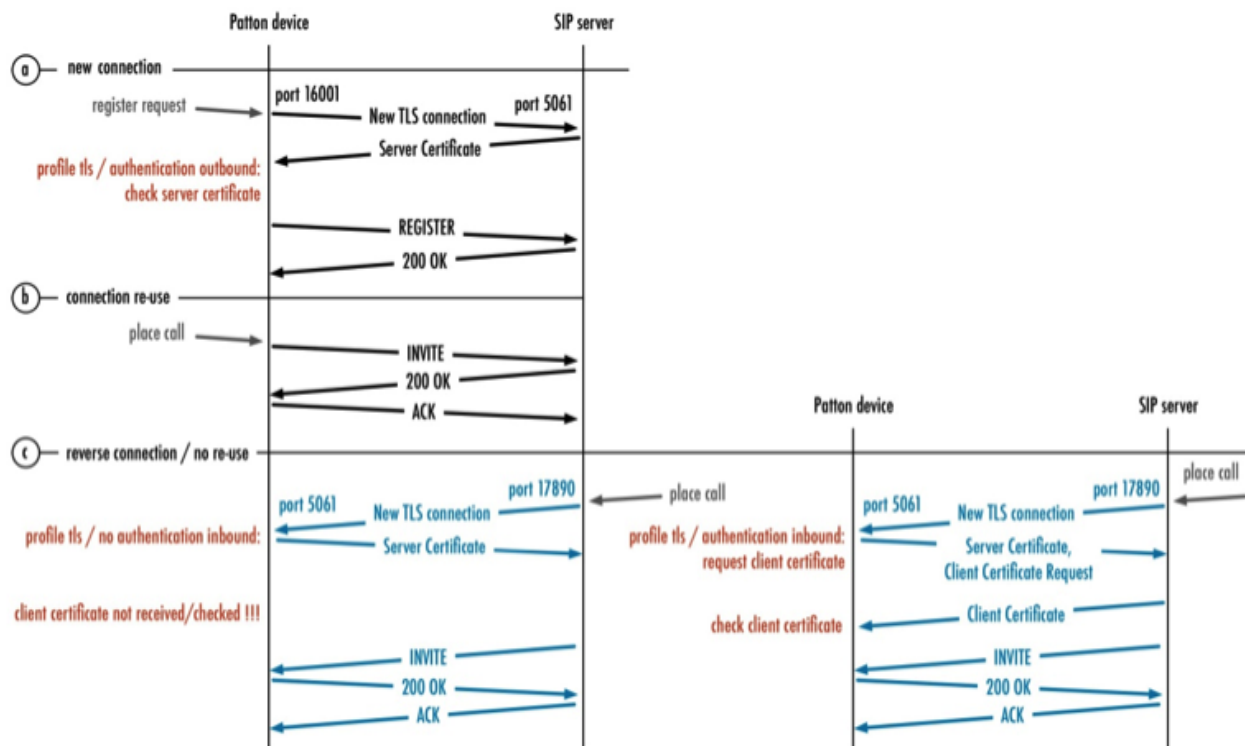


Figure 110. TLS Connection Establishments and Re-use in SIP

- A. When the Patton device has to send a request such as a REGISTER, it may open a new TLS connection. Acting as a TLS client in this scenario, the Patton device uses an ephemeral port (e.g. 16001) and makes a connection to the default TLS port 5061 of the server (or any other port configured in the Request URI). The server then sends back its certificate. The authentication outbound command in the TLS profile decides whether or not our device verifies this certificate (i.e., whether or not it (i) compares the IP address or hostname of the server to the *canonical name* field of the certificate and (ii) verifies that the TLS profile lists in the list of trusted certificates the root CA of the received certificate of the server). Once the TLS handshake completed successfully, the REGISTER request and the corresponding response are sent over this established connection – no new TLS connection has to be opened for a response.
- B. If the Patton device wants to send another request (such as an INVITE or re-REGISTER request) to the same SIP server, it re-uses the TLS connection and therefore does not verify the server's certificate again.
- C. If the SIP server attempts to establish a call to the Patton device at a later point in time. The SIP server cannot re-use the connection, because the Request URI of the INVITE request requires the call to be connected to our local port 5061. Thus the SIP server creates a new TLS connection (blue). The Patton device is the server side for this TLS connection and the authentication inbound command in the TLS profile decides whether or not we request and verify the certificate of the TLS client (which is the SIP server in this case). Note that such a new connection is not only opened for a new call, but also for a new request during an existing call, such as a re-INVITE or a BYE request to renegotiate or drop the call, respectively.

Two Scenarios of TLS Authentication Vulnerability

In the above scenario, assume that **authentication outbound** is enabled but **no authentication inbound** has been configured (left part of figure 110). Let us focus on scenario c where the SIP server places a call towards our device. The SIP server certificate (= TLS client) is not verified since it could be a security issue.

Another, opposite scenario is to have enabled **authentication inbound** but **no authentication outbound** configured. For scenario C, where the SIP server places a call towards our device, we would verify the certificate of the SIP server (=TLS client). The SIP server certificate is no verified again if the server did not act according to the SIP standard and re-used an existing TLS connection we established before. Remember that the command **no authentication outbound** does not check the certificate for outbound TLS connections.

The recommendation is enabling both **authentication outbound** and **authentication inbound** together on the TLS profile whenever possible. The reason for separating the commands was to maximize compatibility in networks where the client does not have its own certificate.

The table below summarizes the vulnerability of the installation due to TLS connections being re-used according to the SIP standard:

Authentication Incoming	Authentication Outgoing	Vulnerability
Enable	Enable	None
Disable	Enable	Connection re-use outgoing
Enable	Disable	Connection re-use oncoming
Disable	Disable	Intentional no authentication

Conditions for the Patton device to Re-use Connections

In order to be precise and complete, this sub-section defines the exact behavior of our SIP application in terms of re-using TLS connections:

- A new outgoing TLS outgoing is opened in the following case:
- SIP Request of a new SIP transaction (e.g. a re-INVITE or BYE request) needs to be sent over TLS, and
- There isn't an opened TLS connection that matches the following parameters:
 - The remote IP address of the connection is equal to the destination IP address of the request,
 - The remote port of the connection is equal to the destination port of the request, and
 - The local IP address of the connection is equal to the bound IP address of the gateway where the request is to be sent.

An existing TLS connection is re-used in the following case:

- The device sends an answer to a incoming request over TLS, or
- The device sends a request over TLS that belongs to an incoming or outgoing transaction, or
- The device sends a SIP Request for a new SIP transaction (e.g. a re-INVITE or BYE request) if there already exists an open TLS connection that matches the following parameters:
 - The remote IP address of the connection is equal to the destination IP address of the request,
 - The remote port of the connection is equal to the destination port of the request, and

- The local IP address of the connection is equal to the bound IP address of the gateway where the request is to be sent.

Caveat for TLS Profile

The TLS profiles have a limitation in distinctive usage. In certain circumstances, a wrong TLS profile is used for an outgoing connection.

The condition in which this issue was observed is:

- Having multiple SIP gateways with different TLS profiles
- The SIP gateways are bound to the same IP address
- A transaction-forming request needs to be sent over one of these gateways

In all other cases there is no issue at all, especially when one of these conditions is met:

- Having only one gateway
- Using the same TLS profile on all gateways
- Having all gateways bound to different IP addresses
- The TLS connection is opened in incoming direction

We plan to remove this limitation in a future software version.

Showing TLS profile

For debugging purposes, there are two commands that can be entered to take a closer look at the TLS profile internals.

Mode: Administrator executable

Step	Command	Purpose
1	node# show profile tls	Shows a list of all TLS profiles.
2	node# show profile tls <name>	Shows the specific TLS Profiles ready to be used in SIP.
3	node# show profile sip-tls	Shows a list of all TLS profiles ready to be used in SIP.
4	node# show profile sip-tls <name>	Shows the specific TLS profiles ready to be used in SIP, separated into client side and server side.

Troubleshooting

Listed below are common problems that will require a troubleshooting procedure:

- check license
- check time
- check certificates
- check TLS profile status

- check sip-gateway status
- check server name

Check License

In order to use TLS or SRTP for SIP a license has to be installed on the Patton device: “sip-tls-srtp”. This can be verified by showing all currently installed licenses by executing **show system licenses**. Verify that “IP TLS and SRTP” line is displayed.

```
node# show system licenses
Software Licenses
=====
                        SIP TLS and SRTP
```

Check time

The time settings can be verified by executing the command **show clock**. It is important to have set the correct time for using TLS. The interesting entry here is “UTC Time”, and the entry “Default Offset” must be set correctly.

```
node# show clock
System Time Information
=====
DST Enabled           : false
Default Offset       : +02:00
Local Time           : Monday September 09 2013 08:30:02
UTC Time             : Monday September 09 2013 06:30:02
Clock                : 2013-09-09T08:30:02
```

An example of how to set the correct time:

```
node (cfg)# clock local default-offset +02:00
node (cfg)# clock set 2013-09-09T08:30:00
node (cfg)#
node (cfg)# show clock
System Time Information
=====
DST Enabled           : false
Default Offset       : +02:00
Local Time           : Monday September 09 2013 08:30:02
UTC Time             : Monday September 09 2013 06:30:02
Clock                : 2013-09-09T08:30:02
```

Check Certificates

The validity period of certificates can be examined by **show pki:certificates/<name>** or **show pki:trusted-certificates/<name>**

```
node# show pki:certificate/MY_CERT
Certificate:
  Data:
    Version: 1 (0x0)
    Serial Number:
      f8:9e:12:de:f9:a9:07:5d
    Signature Algorithm: sha1WithRSAEncryption
    Issuer: L=Bern, CN=192.168.10.10
    Validity
      Not Before: Jul 12 20:48:49 2013 GMT
      Not After : Jul  3 20:48:49 2014 GMT
    Subject: L=Bern, CN=192.168. 10.10
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      Public-Key: (512 bit)
      Modulus:
        00:e8:f8:07:3c:77:b4:c3:37:d6:58:11:66:fb:3d:
        82:52:bf:5c:b3:d9:53:e8:eb:9b:3e:8a:e6:23:ab:
        c7:b3:f0:6c:12:e3:97:63:42:ab:86:f2:13:f2:f1:
        d7:24:04:af:a1:0e:3d:22:03:a9:60:73:aa:d4:cd:
        70:82:50:08:55
      Exponent: 65537 (0x10001)
    Signature Algorithm: sha1WithRSAEncryption
    7e:ab:68:fb:d1:50:ce:5e:69:80:85:1f:73:84:ba:03:6a:76:
    6e:a9:ad:e3:27:08:67:cd:4d:7b:fe:51:56:4b:50:9b:37:90:
    25:f0:a2:78:73:33:84:31:09:08:75:b0:18:ad:7e:bc:8d:f5:
    8d:6b:60:20:32:61:ac:c6:10:9d
```

Check TLS Profile Status

The loading status of the TLS profile as it is seen by our SIP application can be examined by executing the command **show profile sip-tls <name>**. It is important here to check that the “Client status” is shown as “Client successful configured” and the “Server status” reported as “Server successful configured”.

If that is the case the following internal checks have been passed successfully:

- The loaded certificates are syntactically valid
- The dependency of the own-certificate chain is correct
- The own-certificate has really been derived from the configured private key

In addition there is a line for each certificate with the status of the validity period, which should be shown as “OK”. The system time is within the validity period of that certificates at showing time.

```
node# show profile sip-tls DEFAULT

SIP-TLS Profile: DEFAULT

=====

Used:                               by 1 module(s)
Client:                              enabled
Client status:                       Client successful configured
Server:                              enabled
Server status:                       Server successful configured

Client Context:
-----

TLS V1 Protocol:                     enabled
SSL V3 Protocol:                     enabled
Compression:                         enabled
Peer Authentication:                 disabled
Ciphers:                             ALL
Validity of Own Certificates:        1
    Own Certificate 1:                OK
Validity of Trusted Certificates:    2
    Trusted Certificate 1:            OK
    Trusted Certificate 2:            OK

Server Context:
-----

TLS V1 Protocol:                     enabled
SSL V3 Protocol:                     enabled
Compression:                         enabled
Peer Authentication:                 disabled
Ciphers:                             ALL
Validity of Own Certificates:        1
    Own Certificate 1:                OK
Validity of Trusted Certificates:    2
```

Check SIP Gateway Status

The status of the SIP gateway can be examined by executing the command **show context sip-gateway detail 4**. Note the “Used TLS profile” for TLS and the status of the transport binding TLS.

```
node # show context sip-gateway detail 4

Gateway: GW_SIP
=====

Active:                               yes
Number of connected calls:             0
Number of ongoing calls:               0
Accumulated number of calls:           0
Used TLS profile:                       DEFAULT
Bound Location Service:                 PATTON

SocketInterface: GW_SIP.MAN
-----

State:                                  Opened

Transport binding:                      UDP
State:                                  Opened
IP-Address:                             192.168.10.10
Port:                                    5060

Transport binding:                      TCP
State:                                  Opened
IP-Address:                             192.168.10.10
Port:                                    5060

Transport binding:                      TLS
State:                                  Opened
IP-Address:                             192.168.10.10
Port:                                    5061

.....continuing
```

Check Server Name

When authenticating outbound while the TLS connection is opened from this device, the common name in the server certificate must match the host name of the request URI. To verify if this check fails or is successful for the TLS connection creation, the user will need to trace.


```
node # debug all-sip detail 5
```

Execute a test-call to trace the sip-signaling and the server-name check during the TLS handshake. Keywords to look for in the trace are:

- TLS handshaking request arrived
- TLS handshake approval for client connection

```
[DBG] [STACK] TLS handshaking request arrived
# [DBG] [STACK] bIsClient: yes
# [DBG] [STACK] bPeerAuth: yes
# [DBG] [STACK] Number of certificate(s) in the chain: 1
# [DBG] [STACK] Certificate 0
# [DBG] [STACK] Version: EVERSION_V3
# [DBG] [STACK] Serial:
# [DBG] [STACK] B7 0A E7 84 40 91 2A 2C
# [DBG] [STACK] Not Before: 2000/6/2 23:40:15
# [DBG] [STACK] Not After: 2999/10/3 23:40:15
# [DBG] [STACK] rstrPeerHostname: 172.16.40.7
# [DBG] [STACK] No SubjectAlternateName Extensions in certificate
# [DBG] [STACK] No CommonName in certificate
# [DBG] [STACK] TLS handshake approval for client connection: NOT
APPROVED, PeerHostname does not match CommonName or SubjectAltName
```

There are three possible outcomes:

1. **A handshaking request visible and not approve.** If the TLS handshake request was not approved there wasn't a SubjectAlternateName or CommonName in the server certificate received which matched the PeerHostName.
 - Certificate contains no SubjectAlternateName or CommonName: The servers certificate needs to be created new with specifying at least one of these parameters.
 - Certificate contains a SubjectAlternateName or CommonName which does not match the PeerHostName: change address-translation in SIP call-control interface to match the resulting request URI of the INVITE to the SubjectAlternateName or CommonName.
2. **A handshaking request visible and approved.** If the TLS handshaking request was approved, continue troubleshooting by capturing Wireshark traces of the TLS connection establishment after getting a TLS handshake request. Probably the TLS connection establishment was successful and the failure happens later in normal SIP signaling
3. **No handshaking request visible:** If the section is not visible, continue troubleshooting by capturing Wire-shark traces of the TLS connection establishment before getting a TLS handshake request.

Note If the SubjectAlternateName or CommonName in the certificate is a host name that needs to be resolved through DNS, the PeerHostName needs to be a DNS name too. The same applies for IP-addresses; if the SubjectAlternateName or CommonName in the certificate is an ip-address, the Peer-HostName needs to be an ip-address as well. IP-addresses do not match the DNS names and vice versa.

Secure Real-Time Protocol (SRTP) Configuration

SRTP provides security, confidentiality, message authentication and replay protection for RTP and RTCP packets. This is achieved by generating a master key for each RTP stream, which is used to encrypt and decrypt the RTP packets. The keys are exchanged in the SDP content of the SIP signaling messages.

To protect the integrity of the SRTP master keys, securing the SIP signaling traffic is required too. This can be achieved by using TLS. Although not recommended it is still possible to use SRTP without TLS.

SRTP configuration task list

To configure the SRTP, perform the tasks in the following sections.

- License
- Information about using DSP channels
- Configure secure call signaling
- Configure RTP security
- Configure optional protection against key compromising through forking
- Configure optional protection against key compromising through call forward

License

The usage of a Patton device for doing SRTP, URI scheme sips and TLS are licensed. It is a precondition to have the license “sip-tls-srtp” installed on the Patton device before being able to perform all configuration steps.

Information about using DSP channels

A call encrypting its media-stream with SRTP always needs a DSP channel to perform the encryption of the outgoing SRTP stream and the decryption of the incoming SRTP stream. Therefore SRTP cannot be used on devices without DSP. The total number of DSP channels needed for one call depends on the involved protocols.

- For calls SIP to ISDN, SIP to FXS, SIP to FXO and vice versa there is always **one DSP channel** needed to do SRTP. That is the same number as when doing RTP without encryption.
- For calls SIP to SIP there are always **two DSP channels** needed to do SRTP. That is the same number as when doing transcoding from one codec to another codec.

When doing SRTP there is not the same possibility to save DSP channels in a SIP to SIP call as in a call without SRTP. The Patton device terminates always the SRTP streams.

Following an overview of using DSP channels in SIP to SIP calls:

RTP Security on SIP session toward A	RTP Security on SIP session toward B	Same or different codes on sessions	Used SDSP Channels
Unsecure RTP	Unsecured RTP	same	0
Unsecure RTP	Unsecure RTP	different	2
Unsecure RTP	SRTP	same	2
Unsecure RTP	SRTP	different	2

RTP Security on SIP session toward A	RTP Security on SIP session toward B	Same or different codes on sessions	Used SDSP Channels
SRTP	SRTP	same	2
SRTP	SRTP	different	2

Configure secure call signaling

The overall security of the SRTP master keys depends on having the sip signaling secured. Therefore, using the sips URI scheme is encouraged because it does force the sip signaling to be secured with TLS until reaching the destination domain. Using the sips URI scheme requires having TLS transport configured for SIP. TLS can be configured and used without URI scheme sips. But consider: TLS secures signaling only until the next hop and if the next hop is a proxy it could forward SIP signaling messages in an unsecured way. See chapter 58, “SIP Security” on page 740 for details about TLS and URI scheme.

Some of the RTP security commands modes are depending on the security of the call signaling. A call is considered as secure signaled if and only if the following two conditions are met.

- The URI scheme of the request URI is equal to “sips”
- The transport used for signaling is TLS

The reasoning why it is not enough to have TLS as transport is twofold. First TLS guarantees secure signaling only until the next hop and in case of a proxy the SRTP keys could be forwarded in an insecure manner toward the device terminating the SRTP stream. The second reason is the sequence of SDP offer construction and the selection of the transport protocol for signaling. The construction of the SDP offer happens first, for which we need to know if we want to offer SRTP. At this time the used URI scheme is known, but the finally selected transport could potentially change in the case of the URI scheme sip. It could be that a message is meant to be sent over an unsecured TCP transport, fails to be sent over TCP, and is finally sent over the secure transport TLS as a fallback.

Configure RTP security

There are 5 different modes how to support or force usage of SRTP:

- Disabled
- Preferred
- Enforced
- Preferred only for secured call signaling
- Enforced only for secured call signaling

Disabled: We don’t offer SRTP and force the remote device to do unprotected RTP. If we get an offer with only SRTP it is rejected with a “488 not acceptable here” message.

Mode: Profile voip

Step	Command	Purpose
1	node(pf-voip)[name]# no rtp security	Disables the usage of SRTP at all and enforces to do unprotected RTP. This is the Default setting.

Preferred: We offer both, SRTP and unprotected RTP in the same offering and let the remote device decide on what to use. SRTP is the first in the ordering to indicate our preference for it. Incoming we accept any offer we receive: Only with SRTP, Only with unprotected RTP or offers with both. In the last case we select SRTP in our answer to the remote device.

Mode: Profile voip

Step	Command	Purpose
1	node(pf-voip)[name]# rtp security preferred	Enables SRTP and unprotected RTP. If both are available SRTP is selected.

Enforced: We don't offer unprotected RTP and force the remote device to do SRTP. If we get an offer with only unprotected RTP it is rejected with a "488 not acceptable here" message.

Mode: Profile voip

Step	Command	Purpose
1	node(pf-voip)[name] # rtp security forced	Disables the usage of unprotected RTP at all and forces to do SRTP.

Preferred only for secured call signaling. The behavior depends on the security of the call signaling. Intention is to offer or accept SRTP only when the keys are protected through secure signaling. For secure calls the same rules apply as when the option Preferred would be selected. For non-secure signaled calls the same rules apply as when the option Disabled would be selected.

Mode: Profile voip

Step	Command	Purpose
1	node(pf-voip)[name] # rtp security preferred sips	Enabled the usage of SRTP only for secured calls. Do unprotected RTP for non-secure calls.

Enforced only for secured call signaling: The behavior depends on the security of the call signaling. Intention is to enforce secure media when the keys are protected through secure signaling. For secure calls the same rules apply as when the option Enforced would be selected. For non-secure signaled calls the same rules apply as when the option Preferred would be selected.

Mode: Profile voip

Step	Command	Purpose
1	node(pf-voip)[name] # rtp security forced sips	Enforce the usage of SRTP only for secured calls. Do offer and accept unprotected RTP and SRTP for non-secure calls.

Protection against key compromising through forking

In a forking scenario there is the possibility of a key being compromised. We are party A sending an INVITE containing the SRTP master key A to a proxy. The proxy forks the INVITE to party B and C and thus B and C getting to know key A, which is used to encrypt the SRTP stream toward party A. Now B picks up the call and C is dropped off the call. This results in a call where C knows the key A and can listen to the SRTP stream from B toward A.

To avoid this we as party A can create a new key A2 and send it upon connecting with a re-INVITE message to party B. This guarantees that as soon as both parties are connected and talking C cannot listen anymore to B's speech. The disadvantage is that because we as party A have no possibility to detect all forking cases, we need to generate a new key and re-INVITE upon connecting always. Therefore this behavior can be enabled or disabled.

Mode: Interface sip

Step	Command	Purpose
1	node(if-sip)[name] # renew-srtp-keys-on-connect	Configures to always generate new SRTP master keys upon connect and sending with a re-INVITE. Default is disabled.

Protection against key compromising through call forward

In a call forward scenario there is the possibility of a key being compromised. We are party A sending an INVITE containing the SRTP master key A to party B. Party B forwards the call toward party C by sending a "302 temporarily moved" answer toward us. We send in some cases now the same INVITE request with the same key A to party C. This results in a call where B knows the key A and can listen to the SRTP stream from C toward A. In other cases we generate a new INVITE request with a new key A2 to be sent to party C, which results in a call where party B does not know any key.

The behavior of generating a new key or sending the same key twice depends on the configuration of call reroute and call redirect. We send the same key only if the commands "new-session-after-redirect" and "call-reroute accept" are both disabled, which is the default setting. To ensure the generation of a new key enable either "new-session-after-redirect", "call-reroute accept" or both. Be aware that these commands existed already before and have other effects apart from influencing SRTP key generation. If you are in doubt enable only "new-session-after-redirect".

Mode: Interface sip

Step	Command	Purpose
1	node(if-sip)[name] # new-session-after-redirect	Configures to always generate a new INVITE request from scratch, when being forwarded. With the effect that a new SRTP key is generated too. Default is disabled.

Mode: Interface sip

Step	Command	Purpose
1	node(if-sip)[name] # call-reroute accept	Configures to forward the call toward call-control, when being forwarded. The call could be pushed back to another port or protocol. If the forwarded call is going to SIP again all rules the sip interface which is reached are applied. This could mean no SRTP is done at all. But if SRTP is offered again it is guaranteed to generate a new SRTP master key. Default is disabled.

Chapter 59 **SIP Call-router Services**

Chapter contents

Introduction	772
SIP Conference-service	772
SIP Conference-service Configuration Task List	772
Entering conference-service configuration mode	772
Configuring the call routing destination	772
Configuring the conference server	773
SIP Location-service	773
SIP Location-service Configuration Task List	774
Entering SIP location-service configuration mode	774
Binding a location service	775
Configuring multi-contact behavior	775
Configuring the hunt timeout	775

Introduction

This chapter contains the description of all SIP specific call router services, which are only available if the software includes the SIP component.

SIP Conference-service

RFC4240 describes how to address different services on a media server without additional SIP headers or header parameters. This mechanism makes use of the fact media servers do not manage users, they manage media services. For that reason RFC4240 defines how to use the user part of the Request-URI as a service indicator. The conference-service provides the functionality described in RFC4240 as 'Conference Service'. It builds the Request-URI to address a conference room instance on a media server according to this standard. The user part of the Request URI will start with the service indicator 'conf=' followed by a unique conference room identifier. All calls with the same conference room identifier will attend the same session. The host part of the Request-URI that represents the media server host name has to be configured by the user. The conference room identifier and the called party number will be taken concatenated with the serial number of the device.

This looks as follows: `conf=<called-nbr><serial-nbr>@<media-server>`

SIP Conference-service Configuration Task List

The following section describes how to create a new conference-service and how to enter the configuration mode of an existing one. In addition it describes all commands and sub commands of the conference-service configuration mode. All configuration tasks for a conference-service are listed below.

- Enter conference-service configuration mode (see page 772)
- Configure the call routing destination (see page 772)
- Configure the conference server (see page 773)

Entering conference-service configuration mode

The **service sip-conference** command enters the configuration mode of an existing conference-service or creates a new one with a specified name. It also destroys an existing service by using the no form of the command.

Mode: Context CS

Step	Command	Purpose
1	[name] (ctx-cs)[switch]#[no] service sip-conference <service-name>	Creates/Destroys a SIP conference-service or enters the configuration mode of an existing one.

Configuring the call routing destination

The call routing destination specifies the next interface, table or service to which the current call has to be forwarded. Detailed information about call routing of a call control service can be found in chapter 49, “[Call Router Configuration](#)” on page 561. Because this is a SIP specific service, the user should be aware that the final destination of the configured call routing must be a SIP interface. This SIP interface will then set up the call to the configured conference server. Be aware that the elements in the routing path do not modify the Request-URI because it has been built by this service. Therefore the final SIP interface must not have configured a remote host.

Mode: Service SIP conference

Step	Command	Purpose
1	[name] (svc-sip-conf)[name]#[no] route call dest-interface <name> OR dest-table <name> OR dest-service <name>	Specifies the next call routing destination for an incoming call. The no form of the command deletes the current routing entry.

Configuring the conference server

As described in the introduction, one of the responsibilities of this service is to build the conference server Request-URI according to RFC4240. The user part will be built from the called-e164 property and the serial number whereas the host part is specified by this command.

Mode: Service SIP conference

Step	Command	Purpose
1	[name] (svc-ls)[name]#[no] service sip-conference [name] conference-server <host-name> [<port>]	Specifies the conference server host name and the port. The no form of the command removes the current entry.

SIP Location-service

- Note** To avoid possible name conflicts, two expressions will be declared here.
- Location Service:** Domain based identity data base.
 - Service Location-Service:** A call-route service that accesses the Location Service declared above.

This service is the main consumer of the address bindings (mapping of the users identity to a contact address) deposited in the location service data base (see chapter 48, “[Location Service](#)” on page 540). Address bindings have been entered to location service data base either by inbound registration or manually by the user. An address binding is an entry that describes on which host a user is currently reachable. If a call is routed to this service, it performs a lookup with the requested URI to find the right contact information.

If the call is originated by the PSTN network, then this URI does not yet exist and must be built first with a mapping table. If this is not done, the lookup uses the called-e164 number and checks if a registered identity matches this number. For more information about this process, see section, “[B2B User Agent with Registered Clients](#)” on page 790.

It is also possible a user is registered with more than one contact. This can be if the user registers with its office SIP phone and also with its SIP soft client. In this case, the location-service's behavior can be configured. On the location-service, no routing command is available. The call will automatically be routed to that SIP interface on which the register request has been received, therefore manually entered address bindings must be provided with a SIP interface as routing destination.

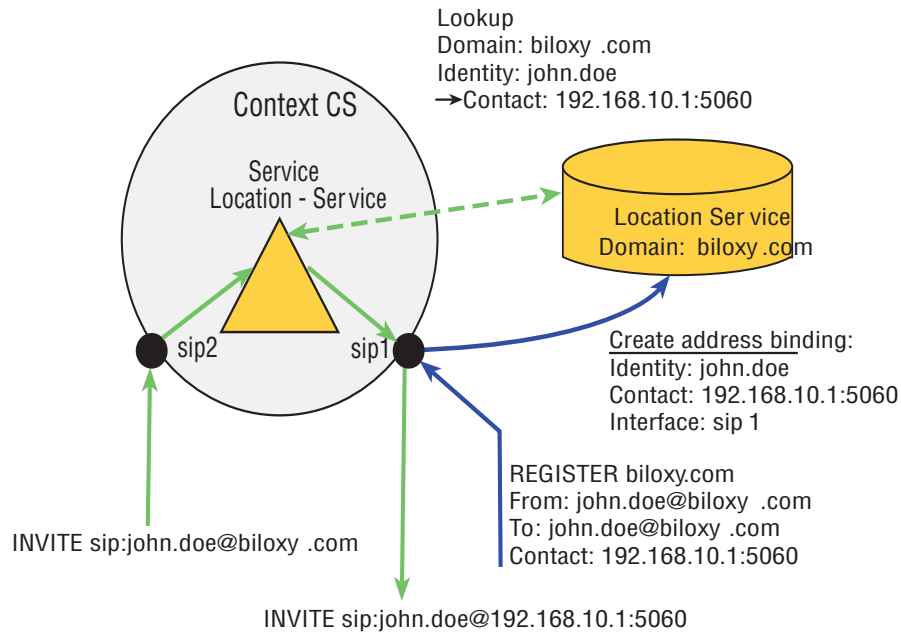


Figure 111. Registration and Lookup

SIP Location-service Configuration Task List

The following section describes how to create a new location-service and how to enter the configuration mode of an existing one. In addition, it describes all commands and sub commands of the location-service configuration mode. All configuration tasks for a location-service are listed below.

- Enter SIP location-service configuration mode (see page 774)
- Bind a location service (see page 775)
- Configure multi contact behavior (see page 775)
- Configure the hunt timeout (see page 775)

Entering SIP location-service configuration mode

The **service sip-location-service** command enters the configuration mode of an existing location-service or creates a new one with a specified name. It also destroys an existing service by using the no form of the command.

Mode: Context CS

Step	Command	Purpose
1	[name] (ctx-cs)[switch]#[no] service sip-location-service <service-name>	Creates/Destroys a SIP location-service or enters the configuration mode of an existing one.

Binding a location service

If a call is routed to the location-service it performs a lookup with the requested URI to the bound location service to find the address bindings. This command is optional because if no location service is bound, all existing location services will be considered to find the right contact information.

Mode: Service SIP location-service

Step	Command	Purpose
1	device(sip-gw)#bind location-service <lsName>	Binds the service to a location service. The no form of the command removes an existing binding. The [no-registration-outbound] command disables registration outbound.(default: enabled)

Configuring multi-contact behavior

There may be cases when a user has registered with several contacts in the location service database to be reachable on several SIP clients. If a call is routed to this service and the lookup in the location service database results in several contacts, this command specifies the behavior for such a case. One possibility is to contact all clients at the same time (distribute) or to contact the clients one after the other (hunt). At time of registration a priority value can be provided. In hunt mode, this priority defines the sequence of contacting the clients. The following modes are available:

- **Hunt:** Contact one registration after the other, depending on the priority. Highest priority first.
- **Distribute:** Contact all registrations at the same time.
- **Distribute and Hunt:** Registrations with the same priority build a distribution group. Hunt over this groups beginning with the highest priority.

Mode: Service SIP location-service

Step	Command	Purpose
1	[name] (svc-ls)[name]#mode {hunt distribute distribute-and-hunt}	Specifies the multi contact behavior of the SIP location-service. Default: distribute-and-hunt

Configuring the hunt timeout

The **hunt-timeout** command is used in 'hunt' mode and 'distribute-and-hunt' mode. It specifies the time that the service must hunt to the next contact if one is not responding. The user must configure this value because it is set per default to zero. This means that the hunting process will stay on the first contact and will never proceed with the next registered contact.

Mode: Service SIP location-service

Step	Command	Purpose
1	[name] (svc-ls)[name]#[no] hunt-timeout <seconds>	Defines the seconds to wait before hunting to the next contact or group of contacts. Default: zero (0)

Chapter 60 **Context SIP Gateway Overview**

Chapter contents

Introduction	778
Context SIP Gateway Configuration Task List	779
Creating a Context SIP Gateway	779
Creating a Transport Interface	779
Configuring the IP Binding	780
Configuring a Spoofed Contact Address	780
Contact-Header	781
Via-Header	781
Nat-Address	781
Binding Location Services	782
SIP Trusted Host Behavior	782
Setting a Traffic Class	782
Configuring Quality of Protection in SIP Authentication	782
Enabling/Disabling the Context SIP Gateway	783
SIP notify check-sync event	783
Troubleshooting	784
Show Status Information	784
Debug Commands	784
Configuration Examples	784
Example 1	784
Example 2	785
Example 3	785
Applications	785
Outbound Authentication	785
Inbound Authentication	787
Outbound Registration	788
Inbound Registration	789
B2B User Agent with Registered Clients	790

Introduction

This chapter provides an overview of the context SIP gateway. The main purpose of a context SIP gateway is forwarding and reception of SIP packets according to a RFC 3261 User Agent. In Trinity, the context SIP gateway represents the interface between the Call-Router (Context CS) and the IP Router (Context IP). It is responsible for dispatching incoming initial SIP requests to the right call control SIP interface and for the determination of the right IP interface for outgoing SIP messages. This is also called *SIP Transport Routing*. Figure 112 shows where the context SIP gateway is located in Trinity and how it interacts with other components.

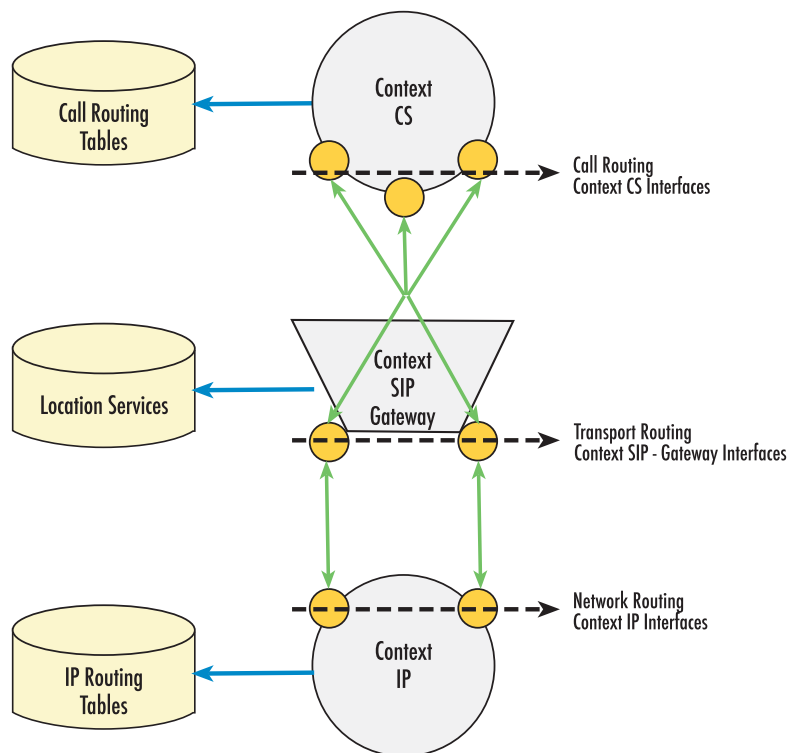


Figure 112. Routing Architecture

There is a relationship between the call control SIP interface and the context SIP gateway. Every call control SIP interface must be bound to a context SIP gateway. One of the responsibilities of this interface is to build Request-URI and to determine a possible outbound proxy. If a proxy is available, it represents the next SIP hop and all outgoing messages will be sent to this host. If no proxy has been configured, the messages will be sent to the Request-URI's host. Depending on these two SIP parameters, the context SIP gateway chooses the right outgoing IP interface. The IP address of the outgoing IP interface will be placed in all SIP and SDP headers that need a direct routable contact address (VIA, Contact).

In the other direction, the context SIP gateway dispatches incoming SIP requests according to the Request-URI and the From-URI. The call control SIP interface has configuration parameters called “remote” and “local” to build the Request-URI, the To-URI and the From-URI for outgoing requests. These parameters will also be used for identifying the best matching call control SIP interface for incoming requests according to the following rule:

1. **From-URI-Host** *equal* **Remote**
and
Request-URI-Host *equal* **Local**
2. **From-URI-Host** *equal* **Remote**
3. **Request-URI-Host** *equal* **Local**
4. No match, the first configured will be taken

For detailed information about call control SIP interface configuration, see chapter 52, “[SIP Interface Configuration](#)” on page 647.

Context SIP Gateway Configuration Task List

The following section describes how to create a new context SIP gateway and how to enter the configuration mode of an existing context SIP gateway. Additionally, it describes all commands and sub commands of the context SIP gateway configuration mode. All configuration tasks for a Context SIP Gateway are listed below.

- Create a context SIP gateway (see [page 779](#))
- Create a transport interface (see [page 779](#))
- Configure the IP binding (see [page 780](#))
- Configure a spoofed contact address (see [page 780](#))
- Bind location services (see [page 782](#))
- Setting a Traffic Class
- Configuring Quality of Protection in SIP Authentication
- Enable/Disable (see [page 783](#))

Creating a Context SIP Gateway

The **context sip-gateway** command enters the configuration mode of a context SIP gateway. If the requested one does not exist, a new one will be created. The no form of the command removes an existing context SIP gateway. This command can be entered without specifying a name. In this case, the default name *sip* will be taken.

Mode: Configure

Step	Command	Purpose
1	<code>[node](cfg)# [no] context sip-gateway [<name>]</code>	Creates/Destroys a context SIP gateway or enter configuration mode.

Creating a Transport Interface

The **interface** command enters the configuration mode of a transport interface. If the requested interface does not exist, a new one will be created. The no form of the command removes an existing transport interface. A maximum of 2 transports interfaces per SIP Gateway can be configured. This limitation is due to the fact, that a Gateway can only be bound to one IP address type (IPv4 or IPv6).

Mode: Context SIP Gateway

Step	Command	Purpose
1	<code>[node](sip-gw)[name]# [no] interface <name></code>	Creates/Destroy a transport interface or enter configuration mode.

Configuring the IP Binding

The following commands specify the ip parameters for the transport layer. A listening socket for UDP and TCP will be created according to the entered parameters. If the port parameter has not been specified default SIP port 5060 will be taken.

Mode: Transport Interface

Step	Command	Purpose
1	<code>[node](sip-if)[name]# [no] bind</code> OR <code>[node](sip-if)[name]#bind ipaddress [ip-context] <ip-interface> <ip-address-label> [port <port>]</code>	Binds the transport interface to the IP layer or removes an existing binding.

Configuring a Spoofed Contact Address

The SIP contact-header and via-header can be independently spoofed using their corresponding option.

Mode: Context sip-gateway/interface

Contact-Header

Step	Command	Purpose
1	device(sip-if)[<name>]# [no] spoofed contact-header (auto manual <host>) [port <port>]	Configures the public visible contact-header. Auto specifies that the NAT address shall be detected and then inserted into the field.

Via-Header

Step	Command	Purpose
1	device(sip-if)[<name>]# [no] spoofed via-header (auto manual <host>) [port <port>]	Configures the public visible via-header. Auto specifies that the NAT address shall be detected and then inserted into the field.

Nat-Address

Step	Command	Purpose
1	device(sip-if)[<name>]# [no] spoofed nat-address (auto manual <ip>) [port <port>]	Configures the public nat address used in sip requests. Showing up in via-header, contact-header and SDP content. Auto specifies that the NAT address shall be detected and then inserted into the field.

Note Public NAT addresses are detected by sending http requests to checkip.dyndns.org, port 80.

If the device is located behind a NAT, this command can be used to provide the Contact (SIP/SDP) and VIA headers of outgoing requests with the public ip address. The IP address may either be specified directly or the address of the NAT may be detected. The port parameter in the spoofed commands is optional. If not specified in the spoofed command, the same port as configured in the context SIP gateway interface is taken. If no port is specified in the interface either, then no port is explicitly signaled which should result in the use of the default port.

Note The options contact-header and via-header can be used together, while the option nat-address can only be used alone.

Mode: SIP Interface

Step	Command	Purpose
1	[node](sip-if)[name]# [no] spoofed { nat-address } { auto manual } [<ip address>]	Applies a spoofed address to the via-header, contact-header and sdp-contact fields to the SIP interface. Alternatively, <i>auto</i> specifies that the NAT address shall be detected and then inserted into the field.

Note Public NAT addresses are detected by sending http requests to checkip.dyndns.org, port 80.

Binding Location Services

The **bind location-service** command binds predefined location services to the context SIP gateway. The no form of the command removes a bound location service. All bound location services define the domains and identities for the context SIP gateway. Identity features, such as outbound registration, inbound registration, and authentication, may or may not be executed, depending on the specific configuration for the identity. Also, they provide transport properties, like Proxy or Traffic Class, and media configuration parameters, like VoIP Profile, SIP Profile or Tone Profile. For more information about configuration of location services and identities, see chapter 48, “Location Service” on page 540.

Mode: Context SIP Gateway

Step	Command	Purpose
1	<code>[node](sip-gw)[name]# [no] bind location-service location-service</code>	Add a location service binding to this context SIP gateway or removes an existing one.

SIP Trusted Host Behavior

In context CS's SIP interfaces, SIP hosts can be set to trusted. By default the SmartNode responds with a “SIP/2.0 503 Service Unavailable” to SIP requests received from hosts which are not trusted. With the [no] form of the command below, the SmartNode will not send any message to untrusted hosts.

Mode: context sip-gateway

Step	Command	Purpose
1	<code>[node](sip-gw)[sip]# [no] answer-untrusted-hosts</code>	Enables "503 Service Unavailable" response for untrusted hosts Default: enabled

Setting a Traffic Class

This feature allows you to directly configure the SIP application with a specific traffic-class. All local generated SIP packets will automatically be marked with the configured traffic-class without the need of applying specific classification rules.

Mode: context sip-gateway

Step	Command	Purpose
1	<code>[node](sip-gw)[name]#traffic-class <tc></code>	Specify specific traffic-classes for the SIP application. Default: local-default

Configuring Quality of Protection in SIP Authentication

For inbound authentication, when the context sip gateway is challenging a request from a user, the quality of protection can be configured. There are four options:

- **none:** This is the default setting and no quality of protection is required. For backward compatibility the request-digest is calculated according RFC 2069.

- **auth:** The quality of protection is set to “Authentication”. The request-digest is calculated according RFC 2617.
- **auth-int:** The quality of protection is set to “Authentication with integrity protection”. The request-digest is calculated according RFC 2617 and the entire body of the message is part of the request-digest calculation.
- **both:** The quality of protection can be either “Authentication” or “Authentication with integrity protection”.

Mode: Context SIP Gateway

Step	Command	Purpose
1	<code>[node](sip-gw)[name]# quality-of-protection (none auth auth-int both)</code>	Configures the quality of protection for inbound authentication. Default: none

Enabling/Disabling the Context SIP Gateway

The **shutdown** command disables the context SIP gateway. The **no shutdown** command enables the context SIP gateway.

Mode: Context SIP Gateway

Step	Command	Purpose
1	<code>[node](sip-gw)[name]# [no] shutdown</code>	Disables the Context SIP Gateway. The no form of the command enables the Context SIP Gateway.

SIP notify check-sync event

It is possible to trigger auto-provisioning with a SIP NOTIFY request. Basically the SIP gateway has to be configured to accept the related packet which will trigger an action script event when the proper information arrived. Finally the action will execute the configured provisioning. The notify packet has to contain the following information:

- A required “Event” header which must be “check-sync”
- An optional “reboot” parameter can be set inside the “Event” header. The possible values are: “true” or “false”. If this parameter is missing then true will be assumed.

Mode: context sip-gateway

Step	Command	Purpose
1	<code>[node](sip-gw)[GW_SIP]#[no] notify check-sync accept</code>	Configures whether the notify check-sync packet will be accepted or not. Default: disabled

Mode: actions

Step	Command	Purpose
1	<code>[node](act)# rule RULE</code>	Defines the rule under which the condition has to be checked.

Step	Command	Purpose
2	<code>[node](act)[RULE]# condition sip <gateway: gatewayname> { NOTIFY_CHECK_SYNC _NORELOAD NOTIFY_CHECK_SYNC _RELOAD }</code>	Defines the condition under which the action has to be executed.
3	<code>[node](act)[RULE]# action "provisioning execute <pf-name>"</code>	Sets what should happen at the end of provisioning.

Mode: profile provisioning

Step	Command	Purpose
1	<code>[node](pf-prov)[<pf-name>]# [no] activation reload { deferred graceful immediate }</code>	Configures whether the notify check-sync packet will be accepted or not. Default: disabled

Troubleshooting

Show Status Information

Mode: Administrator Execution

Step	Command	Purpose
1	<code>[node]#show context sip-gateway [gw-name] [detail <level>]</code>	Displays status and configuration information about a context sip-gateway and its bound resources like the call-control sip interfaces.

Debug Commands

Mode: Administrator Execution

Step	Command	Purpose
1	<code>[node]#debug sip-datapath [detail <level>]</code>	Logs information related to the media channels
2	<code>[node]#debug sip-registration [detail <level>]</code>	Logs information about user registration activities
3	<code>[node]#debug sip-signaling [detail <level>]</code>	Logs call signaling related information
4	<code>[node]#debug sip-transport [detail <level>]</code>	Logs all SIP messages sent or received over the IP network

Configuration Examples

Example 1

This is the minimal configuration of a working context sip-gateway. It has one transport interface that is bound to the ip address LAN. With this configuration, the context sip-gateway processes all SIP requests addressed to port 5060.

```
context sip-gateway SIP-GW
```

```

interface lan
  transport-protocol udp+tcp 5060
  bind ipaddress ROUTER LAN LAN
context sip-gateway SIP-GW
  no shutdown

```

Example 2

The following example shows a context sip-gateway that is explicitly bound to a LAN-Side and a WAN-Side ip interface. This could be a Back-To-Back User Agent application that interconnects a private SIP environment and a public SIP environment.

```

context sip-gateway SIP-GW
  interface lan
    transport-protocol udp+tcp 5060
    bind ipaddress ROUTER LAN LAN
  interface wan
    transport-protocol udp+tcp 5060
    bind ipaddress ROUTER WAN WAN
context sip-gateway SIP-GW
  no shutdown

```

Example 3

If special features like Outbound Registration, Inbound Registration or Authentication is required, one or more location services must be configured that enables these capabilities in general, for a group of identities or just for a single identity. These location services must be bound to the context sip-gateway that is responsible to manage the location service's domains. For detailed description about location service configuration, see chapter 47, “[Authentication Service](#)” on page 537.

```

context sip-gateway SIP-GW
  interface lan
    transport-protocol udp+tcp 5060
    bind ipaddress ROUTER LAN LAN
  interface wan
    transport-protocol udp+tcp 5060
    bind ipaddress ROUTER WAN WAN
context sip-gateway SIP-GW
  bind location-service SIP_LS_1
  no shutdown

```

Applications

Outbound Authentication

The back-to-back user agent can provide credentials for authentication on another sip user agent or proxy. The username and password used for authentication must be configured in an authentication-service. If one or more realms are configured in the authentication-service, the credentials are only provided to challenges containing one of these realms. If no realm is configured, the credentials are provided to any realm.

In an authentication-service, there can be multiple usernames and passwords. An identity which should authenticate can direct the authentication outbound face to a pair of credentials. There can be multiple identi-

ties using the same credentials. An identity can also point to multiple credentials, but each of these credentials needs to be in another authentication-service with another realm. It is possible to authenticate to multiple realms with multiple credentials at the same time.

If the gateway has to provide credentials for unknown identities or for any identity which belongs to a certain domain, there can be a DEFAULT identity-group configured. The authentication credentials configured in the identity-group DEFAULT are used for any identity in this location-service that is not explicitly configured.

```

authentication-service AUTH_PATTON
  realm patton.com
  username hermes password Wh6Xbk9G= encrypted
  username john password Fa0Y9e4L= encrypted

authentication-service AUTH_ANY
  username bob password Co7s3bUp= encrypted

location-service PATTON
  domain voice-cloud.com
  domain patton.com

identity-group default
  authentication outbound
    authenticate 1 authentication-service AUTH_ANY username bob

identity 400
  authentication outbound
    authenticate 1 authentication-service AUTH_PATTON username hermes
    authenticate 2 authentication-service AUTH_ANY username bob

identity 500
  authentication outbound
    authenticate 1 authentication-service AUTH_PATTON username hermes

identity 600
  authentication outbound
    authenticate 1 authentication-service AUTH_PATTON username john

```

If the gateway needs to provide authentication credentials on a sip request, the following procedure takes place:

1. Determine the location-service which provides credentials. The domain of the location service must match the host part of the from-uri and the location-service is bound to the context sip-gateway which sends the request.
2. Determine the identity which provides credentials. The name or the alias of the identity must match the user part of the from-uri. If there is no identity that matches and an identity-group with the name DEFAULT is configured, the identity-group DEFAULT is taken.
3. Determine the authentication-service which provides credentials. The authentication entries of the taken identity or identity-group are searched for an authentication-service that matches exactly the realm requested in the answer to our request. Then this authentication service is taken. If no match was found, an authentication service with no realm configured is taken.

4. Determine the authentication username which provides credentials. If the authentication entry of the identity which configures the taken authentication service has also configured a username this username is taken. If there is no username configured the name of the identity is taken as username.
5. Take the credentials in the authentication service with the according username and provide username and password for re-issuing the request.

If one of these steps has no result and fails, authentication is not possible for that request.

Inbound Authentication

The back-to-back user agent can challenge another sip user agent or proxy for authentication credentials. The username and password used for challenges must be configured in an authentication-service. There must be at least one realm configured in the authentication-service. The first realm configured is used for challenging requests.

In an authentication-service, there can be multiple usernames and passwords. An identity which should be challenged can direct the authentication inbound face to a pair of credentials. There can be multiple identities using exactly the same credentials. An identity can also point to multiple credentials, but only the first entry is used for challenging. If an identity points to multiple credentials, any of these credentials are accepted in the answer as long as it is valid for the challenged realm.

If the gateway has to challenge credentials for unknown identities or for any identity which belongs to a certain domain, there can be a DEFAULT identity-group. The challenging credentials configured in the identity-group DEFAULT are used for any identity in this location-service that is not explicitly configured.

```
authentication-service AUTH_PATTON
  realm patton.com
  username kevin password Wh6Xbk9G= encrypted
  username dirk password Fa0Y9e4L= encrypted
  username boss password Q9Gns6Nd4= encrypted

location-service PATTON
  domain patton.com

  identity-group default
    authentication inbound
      authenticate 1 authentication-service AUTH_PATTON username kevin

  identity 400
    authentication inbound
      authenticate 1 authentication-service AUTH_PATTON username kevin
      authenticate 2 authentication-service AUTH_PATTON username dirk

  identity 555
    authentication inbound
      authenticate 1 authentication-service AUTH_PATTON username boss
```

If the gateway receives an incoming request without credentials, the following procedure takes place:

1. Determine the location-service which challenges credentials. The domain of the location service must match the host part of the request-uri and the location-service is bound to the context sip-gateway which handles the request.

2. Determine the identity which challenges credentials. The name or the alias of the identity must match the user part of the request-uri. If there is no identity that matches and an identity-group with the name DEFAULT is configured, the identity-group DEFAULT is taken.
3. Determine the authentication-service which challenges credentials. The first authentication entry of the taken identity or identity-group where a realm is configured is taken.
4. The first realm in the authentication-service is used for challenging the request. If no realm was found no challenging is possible.

If one of these steps has no result and fails, challenging is not possible for that request and the request is accepted.

If the gateway receives an incoming request with credentials, the following procedure takes place:

1. Determine the location-service which challenges credentials. The domain of the location service must match the host part of the request-uri and the location-service is bound to the context sip-gateway which handles the request.
2. Determine the identity which challenges credentials. The name or the alias of the identity must match the user part of the request-uri. If there is no identity that matches and an identity-group with the name DEFAULT is configured, the identity-group DEFAULT is taken.
3. Check credentials. All authentication entries of the taken identity or identity-group are compared with the provided credentials. If one matches the request is accepted.
4. An authentication entry matches if the username and password matches with the configured credentials and one realm in the authentication-service match exactly the realm challenged or the authentication-service has no realm configured.

If one of these steps has no result and fails, the request is treated as a request without credentials.

Outbound Registration

The back-to-back user agent can register identities on a sip registrar. Therefore, a registration outbound face with the register “auto” command is needed. The address of record is built with the domain of the location-service and the name of the identity. The REGISTER request is sent to the configured registrar or, if missing, to the domain configured in the location-service.

If the registration is successful, the registrar forwards requests that are destined to the address of record to the back-to-back user agent.

```
location-service PATTON
  domain patton.com

identity-group DEFAULT
  registration outbound
  register auto

identity 400
  registration outbound
  registrar sip.patton.com
  register auto

identity 555 inherits DEFAULT
```



```
context sip-gateway GW_SIP

sip-interface SIP_WAN
transport-protocol udp+tcp 5060
bind ipaddress ROUTER IF_WAN IF_WAN

context sip-gateway GW_SIP
bind location-service PATTON
```

If an identity is added to a location-service which is bound to a context sip-gateway, the following procedure takes place:

1. Determine if identity should be registered.
 - The location-service containing the identity must have at least one domain configured.
 - The identity must have a registration outbound face configured or inherited.
 - The register command must be set to “auto”.
2. Build the address of record to register. The name of the identity builds the user-part. The first domain configured in the location-service builds the host-part.
3. Build the address of the registrar. The registrar configured in the registration outbound face is taken as request-uri. If no registrar is configured the first domain configured in the location-service builds the request-uri.
4. Build expire header. If a lifetime is configured in the registration outbound face an expire header with the desired lifetime is added.
5. Build contact address to register. The spoofed-contact parameter of the sip-interface in the context sip-gateway through which the REGISTER request is sent is set as contact address. If no spoofed-contact is configured the ip-address and port of the sip-interface in the context sip-gateway through which the REGISTER request is sent builds the contact address.
6. Send the REGISTER request.

If one of these steps has no result and fails, the registration fails. After a certain timeout (which is configurable in the registration outbound face), the request is re-issued.

Inbound Registration

With the according license, the back-to-back user agent can allow registrations from other user agents. Therefore, identities must be configured with a registration inbound face. Contacts to forward requests for this identity can be configured or added dynamically with REGISTER requests.

If the gateway has to accept register requests for unknown identities or for any identity that belongs to a certain domain, there can be a DEFAULT identity-group configured. The configuration of the identity-group DEFAULT with the registration inbound face allows registration for any identity in this location-service that is not explicitly configured.

```
location-service PATTON
domain patton.com

identity-group DEFAULT
registration inbound
```

```

identity 400
  registration inbound
    lifetime default 4000 min 600 max 36000
    contact 172.16.40.22 switch IF_SIP priority 500

context sip-gateway GW_SIP

  sip-interface SIP_WAN
    transport-protocol udp+tcp 5060
    bind ipaddress ROUTER WAN WAN

context sip-gateway GW_SIP
  bind location-service PATTON

```

If the gateway receives an incoming REGISTER request, the following procedure takes place:

1. Determine to which sip interface in the context cs the request should be forwarded. This happens according the same rules as an incoming INVITE is forwarded. Outgoing calls to the registered contacts will pass through the same sip interface as the incoming REGISTER request.
2. Check request-uri. The host part of the request-uri must match a domain of a location-service which has configured imperative “authoritative” and is bound to the context sip-gateway which received the request.
3. Check to header. The host part of the to-uri must match the host part of the request-uri.
4. Check if registration is allowed. There must be an identity in the location-service that match with name or alias the host part of the to-uri or a identity-group DEFAULT with a registration inbound face configured.
5. Create a dynamic identity if the identity to register does not already exist. This happens when the identity-group DEFAULT is configured with a registration inbound face. The dynamic created identity inherits from the identity-group DEFAULT.
6. Add all contacts with expires>0 to the requested identity. The expiration time is taken out of the expire parameter from the contact or from the expire header. This time is adjusted to fit into the configured lifetime boarder. If there is no expires parameter the default expired from the configured lifetime is taken.
7. Remove all contacts with expires=0 from the requested identity.
8. Remove a dynamic identity if the identity contains no more contacts.
9. Return 200 OK with all contacts registered or configured from the requested identity

If one of these steps fails the registration fails and an according message is sent. A registered contact is removed out of the location-service after the expiration time has passed and the registration was not refreshed. The “[SIP Location-service Configuration Task List](#)” (on page 774) in the context cs is able to forward calls to the registered contacts.

B2B User Agent with Registered Clients

The sip-location-service in the context cs is used to route calls according to the contact bindings. Calls can be routed from other services, interfaces or tables to the sip-location-service. The sip-location-service routes the call directly to the sip-interface over which the according contact was registered. If there are some mappings or adjustments of call parameters needed, this has to be done before routing to the sip-location-service. The address-translation of the destined sip interface is the only mapping that happens after the sip-location-service.

If calls from an ISDN, FXS or FXO interface are routed to the sip-location-service, it is necessary to bind a location-service to the sip-location-service because the domain information is needed. An alternative way would be to map a complete sip-uri to the call destination properties.

The mode command in the sip-location-service configures the behavior of the service when multiple contacts are registered for one address of record. In “distribute” mode, the call is distributed to all contacts at the same time. The first phone that picks up receives the call. In “hunt” mode, one contact after each other is called, and the hunt-timeout parameter defines how long to wait until proceeding to the next contact. The contacts with higher priority are hunted first. In “distribute-and-hunt” mode, all contacts with the same priority are distributed together and the hunt searches from the higher priority contacts to the lower priority contacts. If no hunt-timeout is configured, there is no hunting.

```

context cs switch
  interface sip IF_SIP
    bind context sip-gateway GW_SIP
    route call dest-service SERVICE_SIP
    remote patton.com
    local patton.com

  interface isdn IF_ISDN
    route call dest-service SERVICE_SIP

  service sip-location-service SERVICE_SIP
    bind location-service PATTON

location-service INALP
  domain patton.com

  identity-group DEFAULT
    registration inbound

context sip-gateway GW_SIP

  sip-interface SIP_WAN
    transport-protocol udp+tcp 5060
    bind ipaddress ROUTER WAN WAN

context sip-gateway GW_SIP
  bind location-service PATTON

```

If the sip-location-service receives a call, the following procedure takes place:

1. Determine the requested domain. If none of these three possibilities matches the call is dropped.
 - If the call has a destination-uri set the host part of that uri is taken as requested domain.
 - If there is no destination-uri set, but a destination-ip-address, this is taken as requested domain.
 - If there is no destination-uri and no destination-ip-address set, but a location-service bound, the requested domain is taken from there.
2. Determine the requested user.
 - If the call has a destination-uri set the user part of that uri is taken as requested user.
 - If there is no destination-uri set, the destination-e164 is taken as requested user.

3. Determine the location-service. The location-services are checked if one domain matches the requested domain and if the imperative of the location-service is “authoritative”. If a location-service is bound and it does not match for the requested domain the call is dropped.
4. Determine the identity. The location-service is searched for an identity where the name or an alias matches the requested user.
5. Get the registered and configured contacts of that identity. The contacts are sorted according to the priorities.
6. Distribute or Hunt the contacts according the configured mode.

If one of these steps fails, the call fails. It is recommended to configure a hunt-group in front of that service to have a fall-back when the call fails because the requested identity is not registered.

Chapter 61 **ISDN Overview**

Chapter contents

Introduction	794
ISDN Reference Points	794
Possible Patton device Port Configurations	795
ISDN UNI Signaling	795
ISDN Configuration Concept	797
ISDN Layering	797

Introduction

This chapter provides an overview of ISDN ports and describes the tasks involved in configuring ISDN ports in Trinity.

ISDN ports are the physical ISDN connections on the Patton devices. There are two types of ISDN ports:

- The ISDN basic rate interface (BRI), and
- The ISDN primary rate interface (PRI).

A BRI port supports two 64kbit/s B-channels for switched voice or data connections, one 16kbit/s D-channel for signaling and always-on data transfer. BRI ports are sometimes called S0 ports. The related PSTN access service is also called Basic Rate Access (BRA).

The PRI port supports thirty 64kbit/s B-channels, one 64kbit/s D-channel and one synchronization timeslot on a standard E1 (G.704) while supporting 23, 64kbit/s B-Channels and one 64kbit/s D-Channel on a standard T1 (G.703) physical layer.

ISDN Reference Points

The ISDN standards define a number of reference points on the interfaces between the various equipment types on an ISDN access line. [figure 113](#) illustrates these reference points. The understanding of these reference points and where they are located is necessary for the configuration of the devices' ISDN ports.

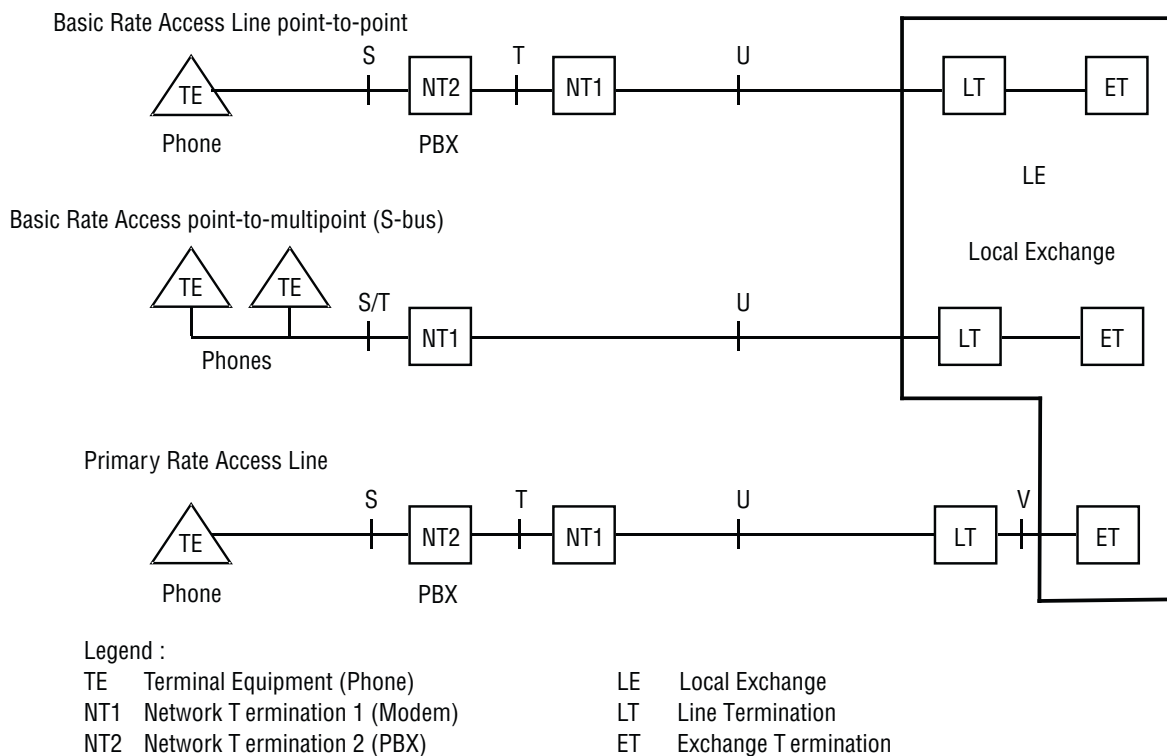


Figure 113. ISDN reference points

The S reference point is on the subscriber interface. This is the typical 4-wire connection between an ISDN phone and an ISDN PBX. Be aware that many ISDN PBX vendors use non-standard proprietary 2-wire interfaces to connect the Terminals to the PBX.

The T reference point is on the trunk interface of a PBX. This is the standard 4-wire interface between the PBX and the network termination unit (NTU) also known as NT1 in standard terminology. The ISDN layer 2 protocol at this point is in point-to-point mode between the NTU and the PBX.

The 4-wire layer 1 specification S and T interfaces is foreseen for in-house installations and carries a maximum of 150 meters.

The S/T reference point is on a point-to-multipoint S-Bus. Here several terminals are connected directly to the same BRI NTU. The S and T reference points are “collapsed”. The NT2 is not represented by any equipment unit.

The U reference point is on the transmission side of the NTU designed to carry the ISDN line over the last mile. For basic rate interfaces this is typically a DSL technology working on legacy copper pairs over a distance up to 12 kilometers. For primary rate lines, DSL, coax and fiber transmission is in use. In most European countries the U interface is not accessible to the subscriber, the operator always provides the NT1. In the US and some other countries the NT1 can be integrated into the NT2, i.e. the PBX is connected directly to the U interface.

The V reference point is typically a y-wire interface between the line card of the public switch and the 2 Mbps transmission equipment which transports the PRI signal over copper (DSL), coax or fiber.

Possible Patton device Port Configurations

The device’s ISDN ports can be configured for connection to S, T, S/T, and V interfaces. Refer to [figure 115](#), which illustrates some of the possible network integration options.

ISDN UNI Signaling

ISDN is a User-Network Interface (UNI) signaling protocol with a user and a network side. The user side is implemented in ISDN terminals (phones, terminal adapters, etc.) while the network side is implemented in the exchange switches of the network operator. Both sides have different signaling states and messages. Trinity ISDN ports can be configured to work as user (USR) or network (NET) interfaces.

A Patton device in some applications does not replace a standard ISDN equipment (PBX or Terminal) but is inserted between an existing NT and PBX. In such cases the device’s ISDN ports are configured to operate the opposite side of the connected equipment as illustrated in [figure 115](#).

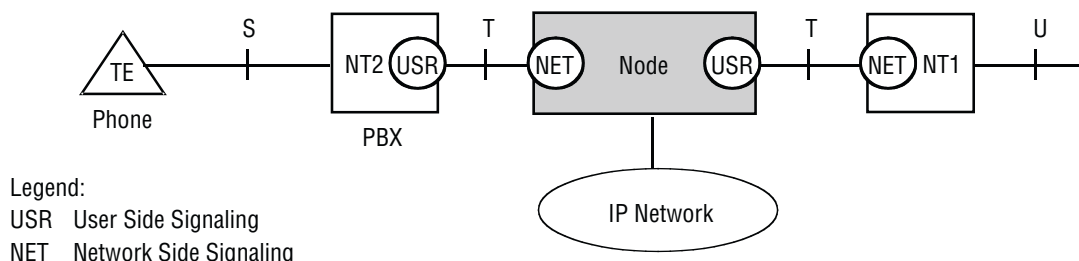


Figure 114. ISDN signaling side

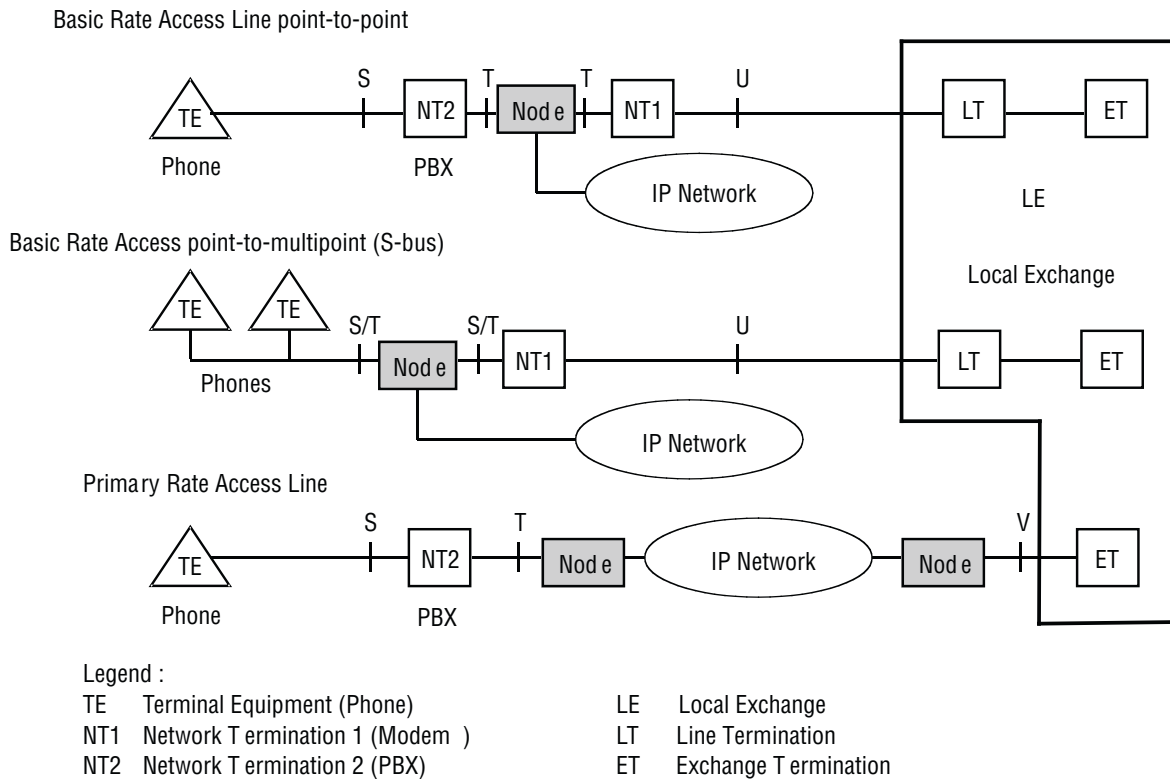


Figure 115. Integration of ISDN access lines



Port activation/deactivation—ISDN ports can be configured while they are active. However they will be internally disabled to modify the configuration and then re-enabled. All active calls on the port are dropped during this process. Configuration changes should only be performed during planned down times.



Reference clock source and synchronization—The Patton device uses a single reference clock source for the synchronization of the 64kbit/s PCM channels on the ISDN ports and in the CS context. This reference clock source can be internal or it can be derived from one of the ISDN ports. If the clock reference is not configured in accordance with the network environment, clock slips and related voice quality degradations can occur. Refer to Chapter 44, “[CS Context Overview](#)” on page 499 on how to configure the reference clock



Connector pin-out and short circuits—Some of the Patton devices' ISDN BRI ports are configurable to operate as network or terminal ports. The pin-out of the sockets is switched according to this configuration. Wrong port configurations, wrong cabling or wrong connections to neighboring equipment can lead to short circuits in the BRI line powering. Refer to the HW installation guide and the port configuration sections below to avoid misconfiguration.

ISDN Configuration Concept

ISDN Layering

ISDN consists of 3 layers. Each layer has its own parameters that need to be configured.

- Layer 1, often called the physical layer, is responsible to transport single bits between two systems. Layer 1 does not guarantee that a message can be transmitted without errors.

Parameters: Clock mode, line codes.

- Layer 2 allows a station to reliably send messages to another station using the D channel. Layer 2 implements flow control, error detection and correction (retransmission) as well as addressing mechanism to direct messages to individual devices.

Parameters: point-to-point or point-to-multipoint mode, network/user side, permanent layer 2 enabled.

- Layer 3 does send and receive application level messages (i.e. call control). It cares for sending broadcast messages and collecting the individual results of the attached devices. It also handles the assignment of the B channels.

Parameters: network/user side, protocol (i.e. DSS1), maximum number of channels.

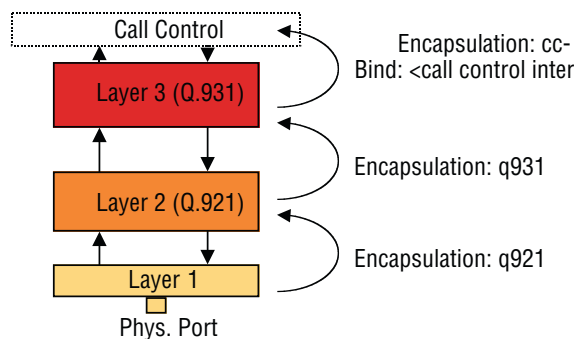


Figure 116. ISDN layering model

The layered model of ISDN is reflected in the configuration by the use of different modes for each layer. The layers are connected by using encapsulations and bindings. The encapsulation defines what the next higher layer protocol will be. On the topmost layer, the binding finally selects a logical interface to connect the port to. For more information how to configure and setup the physical ports for ISDN, please see Chapter 65, “PRI Port Configuration” on page 872. Detailed information about Q.921 and Q.931 configuration are available in Chapter 62, “ISDN Configuration” on page 798.

Chapter 62 **ISDN Configuration**

Chapter contents

Introduction	799
ISDN Configuration Task List	799
Enter Q.921 Configuration Mode	799
Configuring Q.921 Parameters	800
Configuring tei Assignment Procedure on ISDN	800
Configuring Q.931 Encapsulation	801
Enter Q.931 Configuration Mode	801
Configuring Q.931 Parameters	801
Configuring a Channel Identifier on ISDN PRI	803
Configuring Q.931 Application Protocol Encapsulation	803
Debugging ISDN	803
ISDN Configuration Examples	804

Introduction

This chapter describes the configuration of the Q.921 and Q.931 protocol and how to bind the ISDN protocol to an application like the Call Control. To get an overview of the ISDN protocol and the layered configuration model of Trinity, please see Chapter 61, “[ISDN Overview](#)” on page 793. In this chapter it is assumed, that the lower layer on which ISDN will be setup is correctly configured. If ISDN has to run on a TDM port like PRI, please see Chapter 65, “[PRI Port Configuration](#)” on page 872.

ISDN Configuration Task List

Configuring ISDN typically consists of the following tasks:

- Enter Q.921 configuration mode
- Configuring Q.921 parameters (see [page 800](#))
- Configuring Q.931 encapsulation (see [page 801](#))
- Enter Q.931 configuration mode (see [page 801](#))
- Configuring Q.931 parameters (see [page 801](#))
- Configuring Q.931 application protocol encapsulation (see [page 803](#))

Enter Q.921 Configuration Mode

Normally, Q.921 is running as ISDN Layer 2 protocol on a BRI or PRI port. But it is also possible another protocol is using Q.921 as its next encapsulation step and then Q.921 will not be configured out of a port context. That means, Q.921 encapsulation can be configured in different configuration modes. For this reason, the command description below refers to the configuration mode in which Q.921 can be enabled by setting the encapsulation to ‘q921’. This configuration mode is called ‘base-mode’ but it is only an alias for the real mode. Once the encapsulation of q921 has been configured, the Q.921 configuration mode can be entered.

Mode: base-mode

Step	Command	Purpose
1	<code>node(base-mode)[slot/port]#[no] encapsulation q921</code>	Enables/Disables Q.921
2	<code>node(base-mode)]#q921</code>	Enter the Q.921 configuration mode

Configuring Q.921 Parameters

This chapter provides an overview of the Q.921 configuration parameters, their syntax and possible restrictions. In case of ISDN, Q.921 settings apply to both BRI and PRI ports. They are defined in the q921 mode. To use Q921, the lower layer encapsulation must be set to q921.

Mode: q921

Step	Command	Purpose
1	node(q921)[e1t1-(slot/port)]#protocol pp OR node(q921)[e1t1-(slot/port)]#protocol pmp	Specify Q.921 operating mode (Default: BRI: pmp, PRI: pp). The Q.921 protocol running on BRI ports can operate in point-to-point (pp) or point-to-multipoint (pmp) mode. Point-to-multipoint is used to connect multiple terminals to an ISDN S-Bus. In some cases small PBXs are also connected to the public ISDN in point-to-multipoint mode. Point-to-point is typically used to connect PBXs to a public or private ISDN. The Q.921 protocol of PRI ports always runs in point-to-point (pp) mode.
2	node(q921)[e1t1-(slot/port)]#uni-side auto OR node(q921)[e1t1-(slot/port)]#uni-side net OR node(q921)[e1t1-(slot/port)]#uni-side user	Specify the UNI side of the interface (Default: auto) If layer1 clock mode is not defined or set to auto this setting also specifies the clock mode for layer1. NET: clock mode = master USR: clock mode = slave If set to auto the UNI side setting is taken from layer3.
3	node(q921)[e1t1-(slot/port)]#[no] permanent-layer2	Enables the Q.921 permanent activity (Default: disabled). By default, the Q.921 protocol is not enabled permanently, i.e. the first call enables it.

Configuring tei Assignment Procedure on ISDN

This command is a configuration option on ISDN net port to support certain PBXs. During the assignment of a terminal endpoint identifier (tei), the user side generates a random value which allows the net side to distinguish multiple connected users. In the strict configuration, this random value is stored to prevent the tei confusion between the users. In the relaxed configuration, it is not stored, which allows a single connected user device to operate without random values.

Mode: port e1t1 / q921

Step	Command	Purpose
1	node(q921)#[no] strict-tei-procedure	Specifies if the assignment of tei should be strict or relaxed. Default: strict procedure is enabled.

Configuring Q.931 Encapsulation

This command specifies the next protocol or application that has to be attached to the Q.921 protocol. In case of ISDN this will always be the Q.931 protocol.

Mode: q921

Step	Command	Purpose
1	<code>node(q921)[slot/port]# q931</code>	Enables/Disables the next application or protocol. Currently only Q.931 is supported.

Enter Q.931 Configuration Mode

This configuration mode is called here 'base-mode' but it is only an alias for the real mode. Once q931 has been configured, the Q.931 configuration mode can be entered.

Mode: base mode

Step	Command	Purpose
1	<code>[name](base-mode)#q931</code>	Enter the Q.931 configuration mode

Configuring Q.931 Parameters

This section provides an overview of the Q.931 configuration parameters, their syntax and possible restrictions. In case of ISDN, Q.931 settings apply to both BRI and PRI ports. They are defined in the q931 mode. To use Q931, the lower layer encapsulation must be set to q931.

Note QSIG is an ISDN based protocol for signaling between nodes of a Private Integrated Services Network. The formal name of the signaling system by ISO / IEC is PSS1. Both names will co-exist and QSIG will continue to be used as the marketing name.

Mode: q931

Step	Command	Purpose
1	<code>device(q931)[e1t1-(slot/port)]#protocol dss1</code> OR <code>device(q931)[e1t1-(slot/port)]#protocol dms-100</code> OR <code>device(q931)[e1t1-(slot/port)]#protocol ni2</code> OR <code>device(q931)[e1t1-(slot/port)]#protocol ntt</code>	Specify the ISDN layer 3 protocol (Default: BRI: dss1, E1: dss1, T1: ni2) The ISDN layer 3 is the network signaling protocol. Trinity ISDN supports: <ul style="list-style-type: none"> • Euro-ISDN (E-DSS1) • Q.SIG (PSS1) • National ISDN (NI2) • Nippon Telecom NTT for BRI • Nortel Dms-100 for T1 The layer 3 signaling must correspond to the connected ISDN equipment or network.

Step	Command	Purpose
2	device(q931)[e1t1-(slot/port)]# uni-side net OR device(q931)[e1t1-(slot/port)]# uni-side user	Specify the UNI side of the interface. If not defined on layer2 (q921 mode) this setting also specifies the UNI side setting for layer2. Make sure that the device connected to a Patton device ISDN port is operating the opposite side of the configured uni-side.
3	device(q931)[e1t1-(slot/port)]#max-calls <max-calls> [1-32] OR device(q931)[e1t1-(slot/port)]#no max-calls	Limits the total number of concurrent calls on the port. The no form of the command restores the default settings. Note: If the channel-range and max-calls command are used simultaneously, the lower number of channels is the limiting parameter.
4	device(q931)[e1t1-(slot/port)]#channel-range <low> <high> OR device(q931)[e1t1-(slot/port)]#no channel-range	Specify B-channel range to be used on a PRI port (Default: E1: 0-31, T1: 0-23) Limits the time-slots to be used for calls to the range between <i>low</i> and <i>high</i> . This is in some cases required for interoperability with ISDN services that impose the same limitations. Call slots outside the defined range are rejected (busy line). If no range is defined (Default) all 30 E1 (23 T1) time-slots are available for use. The no form of the command restores the default settings.

Step	Command	Purpose
5	device(q931)[e1t1-(slot/port)]# bchan-number- order ascending OR device(q931)[e1t1-(slot/port)]# bchan-number-order ascending-cyclic OR device(q931)[e1t1-(slot/port)]# bchan-number-order descending OR device(q931)[e1t1-(slot/port)]# bchan-number-order descending-cyclic	Specify B-channel allocation strategy (Default: ascending) The numbering mode defines how the available time slots are filled. The cyclic modes use a “round-robin” implementation. The “ascending” and “descending” modes define whether the time slots are filled at the lowest or highest available slot, i.e. ascending means that the lowest available slot is used first, descending always uses the highest available slot first.

Configuring a Channel Identifier on ISDN PRI

If the sending of a channel identifier in the SETUP message on the ISDN PRI user interface is forced, in some cases, the channel identifier information element is not according to the Standard. This behavior can be explicitly disabled. When the sending of the channel identifier is not forced, it is only sent when it is complete and valid.

Mode: q931

Step	Command	Purpose
1	[node](q931)[X/Y]#[no] force-channel-id-in-setup	Forces the sending of a channel identifier in SETUP. Default: Enabled

Configuring Q.931 Application Protocol Encapsulation

This command specifies the next protocol or application has to be attached to the Q.931 protocol. In case of ISDN this will always be the CC-ISDN (Call Control) application. For this case also a binding to a pre-created ISDN interface is necessary. For information about creation and configuration of an ISDN interface please see Chapter 63, “ISDN Interface Configuration” on page 806.

Mode: q931

Step	Command	Purpose
1	node (q931)[e1t1-(slot/port)]# [no] bind interface <if-name>	Bind the Q.931 protocol to an existing call control interface.

Debugging ISDN

The **debug isdn** command may be used for investigating possible Q.921/Q.931 protocol problems or to get call signaling information. The command can be applied to the port on which ISDN is configured and has a further option to switch on or off a specific ISDN layer. Additionally, the ‘**show port [e1t1 or bri]**’ command

can be used to printout information about the current state and statistical information about received and transmitted frames.

Mode: Operator execution

ISDN Command	Purpose
isdn-control	Enables/Disables call-control ISDN control trace
isdn-datapath	Enables/Disables call-control ISDN datapath trace
isdn-signaling	Enables/Disables call-control ISDN signaling trace
isdn-stack-pri-control	Enables/Disables ISDN stack PRI control trace
isdn-stack-pri-l2	Enables/Disables ISDN stack PRI layer 2 trace
isdn-stack-pri-l3	Enables/Disables ISDN stack PRI layer 3 trace
master-server	Enables/Disables Master Server trace
all-isdn	Enables/Disables all ISDN trace

Mode: Operator execution

Debug ISDN Command	Purpose
debug-isdn-control	Enables/Disables call-control ISDN control trace
debug isdn-datapath	Enables/Disables call-control ISDN datapath trace
debug isdn-signaling	Enables/Disables call-control ISDN signaling trace
debug isdn-stack-bri-control	Enables/Disables ISDN stack BRI control trace
debug isdn-stack-bri-l2	Enables/Disables ISDN stack BRI layer 2 trace
debug isdn-stack-bri-l3	Enables/Disables ISDN stack BRI layer 3 trace
debug isdn-stack-pri-control	Enables/Disables ISDN stack PRI control trace
debug isdn-stack-pri-l2	Enables/Disables ISDN stack PRI layer 2 trace
debug isdn-stack-pri-l3	Enables/Disables ISDN stack PRI layer 3 trace

ISDN Configuration Examples

Example: Configuring BRI port as Euro-ISDN interface

The following example shows how to configure port 0/0 as a Euro ISDN interface with user side signaling.

```
172.16.40.71(cfg)#port bri 0 0
172.16.40.71(prt-bri)[0/0]#q921
172.16.40.71(q921)[0/0]#q931
172.16.40.71(q931)[0/0]#uni-side user
172.16.40.71(q931)[0/0]#
172.16.40.71(q931)[0/0]#bind interface bri00
172.16.40.71(q931)[0/0]#exit
172.16.40.71(q921)[0/0]#exit
172.16.40.71(prt-bri)[0/0]#no shutdown
```


Example: being clock slave on uni network interface

The following example shows how to configure both ports of a Patton device with network signaling but receive the clock (via port 0) from the peer. The peer must be configured accordingly, i.e. port 0 as USR/clock master and port 1 NET/clock slave.

```
172.16.40.71(cfg)#port bri 0 0
172.16.40.71(prt-bri)[0/0]#clock slave
172.16.40.71(prt-bri)[0/0]#q921
172.16.40.71(q921)[0/0]#q931
172.16.40.71(q931)[0/0]#uni-side net
172.16.40.71(q931)[0/0]#
172.16.40.71(q931)[0/0]#bind interface bri00
172.16.40.71(q931)[0/0]#exit
172.16.40.71(q921)[0/0]#exit
172.16.40.71(prt-bri)[0/0]#no shutdown

172.16.40.71(cfg)#port bri 0 1
172.16.40.71(prt-bri)[0/0]#q921
172.16.40.71(q921)[0/0]#q931
172.16.40.71(q931)[0/0]#uni-side net
172.16.40.71(q931)[0/0]#
172.16.40.71(q931)[0/0]#bind interface bri01
172.16.40.71(q931)[0/0]#exit
172.16.40.71(q921)[0/0]#exit
172.16.40.71(prt-bri)[0/0]#no shutdown
```

Example: PRI

Assume the scenario as illustrated in [figure 117](#):

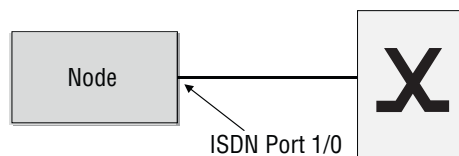


Figure 117. PBX connected to ISDN port 1/0

Configure PRI port 1/0 as clock master. From the Local Exchange timeslots 1 through 20 are available and the total number of concurrent calls shall be limited to 10. Use down-cyclic channel numbering.

```
172.16.40.71(cfg)#port elt1 1 0
172.16.40.71(prt-elt1)[1/0]#q921
172.16.40.71(q921)[elt1-1/0]#q931
172.16.40.71(q931)[elt1-1/0]#uni-side net
172.16.40.71(q931)[elt1-1/0]#max-channels 10
172.16.40.71(q931)[elt1-1/0]#channel-range 1 20
172.16.40.71(q931)[elt1-1/0]#bchan-number-order descending-cyclic
172.16.40.71(q931)[elt1-1/0]#exit
172.16.40.71(q921)[elt1-1/0]#exit
172.16.40.71(prt-el)[elt1-1/0]#no shutdown
```

Chapter 63 ISDN Interface Configuration

Chapter contents

Introduction	807
ISDN Interface Configuration Task List	807
Configuring DTMF Dialing (optional)	808
Configuring an Alternate PSTN Profile (optional)	808
Configuring Ringback Tone on ISDN User-side Interfaces	809
Configuring Call Waiting (optional)	809
Disabling Call Waiting on ISDN DSS1 Network Interfaces	809
Configuring Call-Hold on ISDN Interfaces	810
Enabling Display Information Elements on ISDN Ports	810
Configurable calling party or facility IE on ISDN	810
Configuring Date/Time Publishing to Terminals (optional)	810
Sending the Connected Party Number (COLP) (optional)	811
Enabling Sending of Progress-indicator on ISDN (optional)	811
Defining the 'network-type' in ISDN Interfaces	811
ISDN Explicit Call Transfer Support (& SIP REFER Transmission)	812
ISDN Advice of Charge Support	814
ISDN DivertingLegInformation2 Facility	817
Transmit direction	817
Receive direction	818
T1 Caller-Name Support	818
Caller-Name Facility Format on QSIG	820
Format Examples	820
Tunneling of ISDN UUI1 Information Over SIP	820
Configuring the Message Waiting Indication Feature for ISDN	821
Configuring the Release Tone on ISDN DSS1 Network Interfaces	821
Configuring the ISDN User-side Timer T304	822

Introduction

This chapter provides an overview of ISDN interfaces, and the tasks involved in their configuration. This chapter does not explain the basic configuration steps equal to all CS interfaces. Information about basic interface configuration can be found in the general chapter about CS interface configuration (see Chapter 45, “CS Interface Configuration” on page 521)

An ISDN interface represents the connection of an ISDN signaling channel to the call control. It encapsulates the ISDN layer 3 protocol of an ISDN port's D-channel, allows incoming and outgoing calls on this port, controls its B-channels and provides a set of services.

There is a one-to-one relation between the port and the interface: Only one port can bind to an existing interface, and there must be a port that binds to the interface for the interface to become functional (see figure 118).

An ISDN interface can encapsulate user and network side of the following protocols: DSS1, NI2, NTT. The settings are automatically taken from the port that binds to the interface, and changes on the port are automatically reflected on the interface.

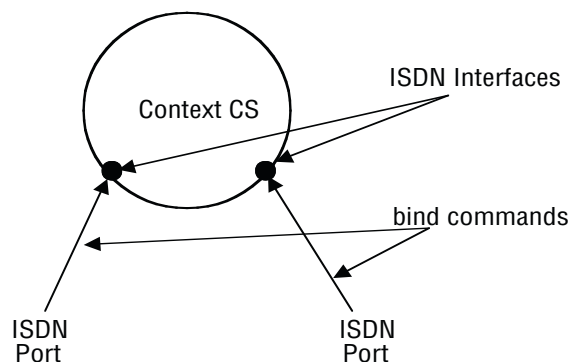


Figure 118. ISDN interfaces on the CS context

ISDN Interface Configuration Task List

This section describes the configuration tasks for ISDN interfaces. There are no mandatory configurations on ISDN interfaces, because all protocol relevant settings are inherited from the port that binds to the interface.

The settings on the interface are those of basic CS interface configuration, as well as settings for interoperability and supplementary services:

- Configuring ringback tone on ISDN user-side interfaces
- Configuring call waiting (optional)
- Disabling call waiting on ISDN DSS1 network interfaces
- Configuring date/time publishing to terminals (optional)
- Enabling sending of date and time on ISDN DSS1 network interfaces
- Defining the 'network-type' in ISDN interfaces
- ISDN explicit call transfer, SIP REFER transmission
- ISDN Advice of Charge support

Configuring DTMF Dialing (optional)

Most ISDN terminals support two modes of call setup: En-bloc dialing and overlap dialing. En-bloc dialing transports the full called party information in the first SETUP message from the terminal. This means that the user must dial the number before going off-hook. Overlap dialing transports the called-party number digit by digit, after the first SETUP message, which contains no called-party information at all. Combinations between en-bloc and overlap dialing are possible.

Most terminals use ISDN *keypad facility* messages to transport digits one-by-one in overlap dialing. But some terminals, especially terminal adapters for analog devices, might transport the digits only using DTMF tones, without associated keypad facility messages.

The **DTMF dialing** command enables the ISDN port for use with the later devices.

Be sure to only use this command when needed. Otherwise, called party information can be corrupted because the digits arrive twice, as keypad facility messages and also as DTMF tones.

Procedure: To enable DTMF dialing

Mode: Interface ISDN

Step	Command	Purpose
1	<code>node(if-isdn)[if-name]#[no] dtmf-dialing</code>	Enables/Disables DTMF dialing Default: disabled

Example: Enable DTMF dialing

The following example shows how to enable DTMF dialing for a given ISDN interface.

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface isdn MyIsdnIf
node(if-isdn)[myIsdnIf]#dtmf-dialing
```

Configuring an Alternate PSTN Profile (optional)

The PSTN profile contains the configuration for data/voice transmission on circuit-switched channels (see Chapter 43, “PSTN Profile Configuration” on page 495). In the case of ISDN interfaces, the PSTN profile applies to the ISDN B-Channels associated with the interface.

There is a PSTN profile named *default*, which always exists in the system. If no different PSTN profile name is explicitly configured on the ISDN interface, the profile *default* is used.

Procedure: To define an alternate PSTN profile for the ISDN interface

Mode: Interface ISDN

Step	Command	Purpose
1	<code>node(if-isdn)[if-name]#[no] use profile pstn profile-name</code>	Defines an alternate PSTN profile to be used for this ISDN interface/Reverts the setting to its default (use profile PSTN <i>default</i>)

Example: Configure an alternate PSTN profile

The following example shows how to replace the PSTN profile *default* of the ISDN interface with the PSTN profile *myprofile*.

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface isdn myIsdnIf
node(if-isdn)[myIsdnIf]#use profile pstn myprofile
```

Configuring Ringback Tone on ISDN User-side Interfaces

If a ring-back tone needs to be played towards the PSTN from an ISDN user-side interface, this can be forced using the following command.

Mode: interface ISDN <if-name>

Step	Command	Purpose
1	[name] (pf-isdn)[if-name]# [no] user-side-ring-back-tone	Enables ringback tone to be played on ISDN user-side interfaces. Default: <i>disabled</i>

Configuring Call Waiting (optional)

The term “call waiting” is used as follows in this context: If the port bound to this interface is configured to be network side, and both ISDN B-Channels are engaged with calls, and there is a new outgoing call over this interface, the interface can:

- Signal the new call to all connected terminals, although both B-Channels are in use. One terminal can then put its current call on hold to accept the new one (putting the call on hold frees its B-Channel).
- Not signal the new call, because there is no B-Channel available. This is the desired behavior particularly if the bound port is part of a hunt-group, and no user terminals are connected.

Default behavior is a), using the command below in the inverted form, behavior b) is selected.

Procedure: To configure call waiting

Mode: Interface ISDN

Step	Command	Purpose
1	node(if-isdn)[if-name]#[no] call-waiting	Enable/disable call waiting feature as described above Default: <i>enabled</i>

Disabling Call Waiting on ISDN DSS1 Network Interfaces

This procedure disables support for call waiting on an ISDN DSS1 network interface.

Mode: interface isdn <if-name>

Step	Command	Purpose
1	[name] (if-isdn)[if-name]# no call-waiting	Disable call waiting.

Configuring Call-Hold on ISDN Interfaces

Normally, the call-hold feature is disabled on ISDN point-to-point links and enabled on ISDN point-to-multipoint links. However, you can manually enable or disable the Call-Hold feature using the following command: The default setting can be achieved using the 'auto' configuration option.

Mode: interface isdn

Step	Command	Purpose
1	node(if-isdn)[if-name]# call-hold {auto enable disable}	Enable or disable the call-hold functionality for an isdn interface. If 'auto' is selected, call-hold is automatically disabled on p2p links and enabled on p2mp links. Default: auto

Enabling Display Information Elements on ISDN Ports

By default no display information elements are sent in ISDN signaling messages. You can enable sending of ISDN Display Information elements in ISDN signaling messages using the following command.

Mode: interface isdn

Step	Command	Purpose
1	node(if-isdn)[if-name]# [no] display emit	Enable sending of the display information element for an isdn interface. Default: disabled

Configurable calling party or facility IE on ISDN

This function defines if called-party numbers starting with „# or „* characters are sent in a keypad-facility info-element or in a calling-party info-element. When the *Keypad-facility* command is inverted, the called-party number is placed in a calling-party info-element.

Mode: context cs switch

Step	Command	Purpose
1	(if-isdn)[if-name]# [no] keypad-facility	Default: enabled

Configuring Date/Time Publishing to Terminals (optional)

ISDN allows to propagate current time and date information from a port configured as network to the connected terminals. You can configure each ISDN interface to propagate the current device system time and date to the connected terminals with the following command:

Procedure: To configure date and time publishing

Mode: Interface ISDN

Step	Command	Purpose
1	node(if-isdn)[if-name]#[no] isdn-date-time	Enable/disable publishing of system time to connected ISDN terminals. Default: disabled

Date and time information can only be contained in the ISDN CONNECT message. This message is only delivered to a terminal when a call from the terminal to the device is made and reaches connected state.

Sending the Connected Party Number (COLP) (optional)

The connected party number information element is normally only inserted into the CONNECT message when it has been verified from the remote side. For certain configurations that depend on the presence of such an information element, it is possible to configure so that the connected party number information element is always inserted.

There are three different configurations for the connected party number information element:

- **disabled:** Never insert the connected party number
- **verified:** Insert the connected party number only if it has been verified. This is the default behavior
- **always:** Insert the connected party number always. If the connected party number is not verified, the called party number is taken as connected party number

Mode: interface ISDN

Step	Command	Purpose
1	node(if-isdn)[if-name]#[no] send-connected-party-number [always verified]	Configures the sending of the connected-party-number. Default: Verified

Enabling Sending of Progress-indicator on ISDN (optional)

In some setup configurations, it is not desired to send a progress-indicator on in the ISDN signaling. Therefore, it is possible to disable the sending of the progress-indicator.

Mode: interface ISDN <if-name>

Step	Command	Purpose
1	[name](if-isdn)[if-name]# [no] send-progress-indicator	Enable/Disable the sending of the information Element progress indicator. Default: enabled

Defining the 'network-type' in ISDN Interfaces

The following command defines the location code to be inserted in ISDN causes code information elements.

Mode: interface ISDN <if-isdn>

Step	Command	Purpose
1	<code>[name] (if-isdn)[if-name]# network-type [international private public transit user beyond]</code>	Defines the type of network to which the system belongs.

ISDN Explicit Call Transfer Support (& SIP REFER Transmission)

Additional call transfer support is enabled by default for ISDN interfaces (BRI ports) by accepting or rejecting explicit call-transfer (ECT) invocations. An ISDN phone that is connected to a BRI port and that has two active calls can send an ECT invocation to connect the two calls inside the device. An ISDN interface can be configured to accept or reject ECT invocations.

Trinity detects calls that are looped internally, i.e. calls that leave the device over the same ISDN interface over which they enter the device. If an internal loop is detected for an ISDN interface bound by an ISDN user port, Trinity sends an explicit call-transfer (ECT) to push back the call to the connected network as soon as the call is connected. An ISDN interface can be configured to emit ECT invocations.

SIP interfaces react similarly to internally looped calls. If a call leaves the device over the same SIP gateway over which it entered the device, Trinity sends a REFER message to one of the remote user agents to transfer the call to the two parties. A SIP interface can be configured to emit REFER messages.

Figure 119 shows an example scenario where a SIP network connects two devices to give a home office (HO) access to a PBX in the central office (CO).

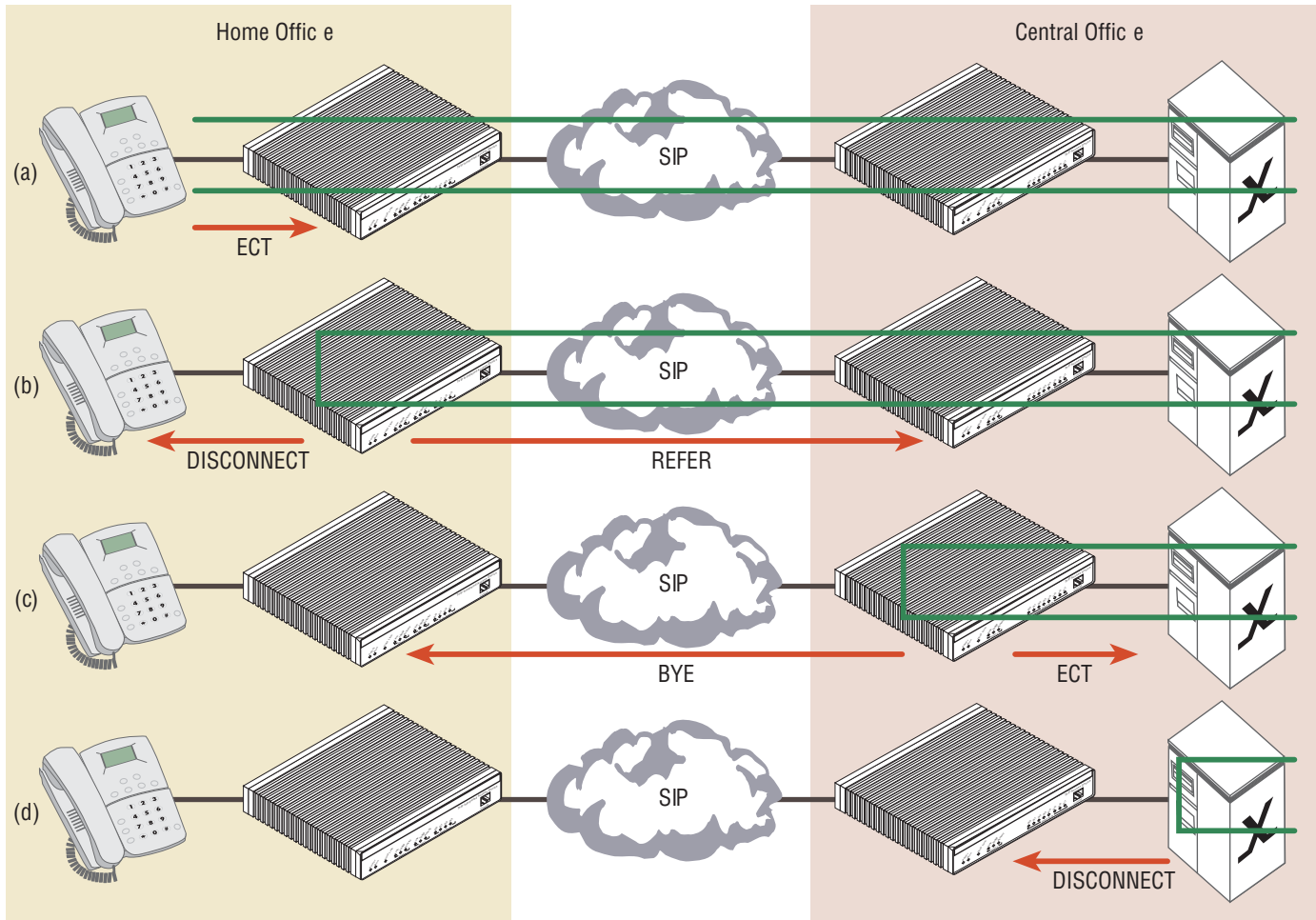


Figure 119. Example SIP network connecting two device to give a home office access to the CO PBX

The phone in the home office has two active calls to other subscribers of the PBX in the central office. The user wants to connect the other two participants and **(a)** sends an explicit call-transfer invocation to the device HO. The device HO internally connects the two calls and sends a DISCONNECT message to the phone for both calls. In a second step **(b)** the firmware on HO detects an internal loop. Both call legs are connected to the same network. In this example, both call legs are handled by the same SIP gateway. The firmware on device HO sends a REFER message to device CO, which connects the two call legs internally and sends a BYE message to the device HO. **(c)** Again the firmware of CO detect an internal loop. This time the call legs are handled by the same SIP interface, connected to the PBX. Since the ISDN port is a user port it sends an explicit call-transfer invocation to the PBX **(d)**, which connects the call and sends the device CO a DISCONNECT message for both calls. During all these push back operations the datapath of the two participants keeps connected.

The push back mechanism over ISDN (using ECT) and SIP (using REFER) works independently of the protocol that invoked the call-transfer.

The push-back mechanism can be configured on each interface separately. Per default push-back is enabled for ISDN and SIP interfaces. You only have to change the configuration if you don't want internally looped calls to

be pushed back to the network. The configuration command **[no] call-transfer accept** configures if an incoming call-transfer request (e.g. ECT or REFER) shall be accepted. The configuration command **[no] call-transfer emit** configures if a call reaching the device over this interface and leaving the device over this interface shall be pushed back to the network, i.e. if a call-transfer request (ECT or REFER) shall be sent.

The following procedure disables the push-back mechanism on the ISDN interface connected to the PBX. No ECT invocation is sent when a call is detected that is looped internally.

Step	Command	Purpose
1	node(ctx-ip)[ctx-name]# interface isdn <if-name>	Go to the ISDN interface, for which you want to disable the push back mechanism.
2	node(if-isdn)[if-name]# no call-transfer emit	Disable the push back mechanism

The following procedure disables the push-back mechanism on a SIP interface. No REFER message is sent when a call is detected that is looped internally.

Step	Command	Purpose
1	node(ctx-ip)[ctx-name]# interface sip <if-name>	Go to the SIP interface, for which you want to disable the push back mechanism.
2	node(if-sip)[if-name]# no call-transfer emit	Disable the push back mechanism

ISDN Advice of Charge Support

The exchange of “Advice of Charge” information is supported between two ISDN interfaces. The charge information can be transmitted and received over SIP. Without configuration changes Trinity tunnels the “Advice of Charge” information from an ISDN user interface to an ISDN network interface. However, you can disable AOC-S, AOC-D or AOC-E separately on each interface.

The network sends tariff information about a call using AOC-S messages at call (S)etup time and during the call when the tariff changes. Then (D)uring the call, the network sends the current charge in AOC-D messages. Finally at the (E)nd of the call, the network sends the total charge in an AOC-E message encapsulated in the DISCONNECT or RELEASE message.

The following procedure disables the reception of AOC messages from the network on an ISDN user interface.

Step	Command	Purpose
1	node(ctx-ip)[ctx-name]# interface isdn <if-name>	Go to the ISDN interface, for which you want to disable AOC
2	node(if-isdn)[if-name]# no aoc-s	Disables the reception of AOC-S messages at call setup time
3	node(if-isdn)[if-name]# no aoc-d	Disables the reception of AOC-D messages during the call
4	node(if-isdn)[if-name]# no aoc-e	Disables the reception of AOC-E messages at the end of the call

AOC is a network option that can be disabled or set to be active for all calls or only on a per-call basis. If your network provider offers AOC on a per-call basis the firmware needs to request AOC information on each out-

going call. The following procedure enables the reception of AOC messages on an ISDN user interface. Additionally the interface sends an AOC activation request for each outgoing call to the network.

Step	Command	Purpose
1	node(ctx-ip)[ctx-name]# interface isdn <if-name>	Go to the ISDN interface, for which you want to enable AOC on a per-call basis
2	node(if-isdn)[if-name]# aoc-s explicit	Enables the reception of AOC-S messages and sends an AOC-S activation request for each outgoing call
3	node(if-isdn)[if-name]# aoc-d explicit	Enables the reception of AOC-D messages and sends an AOC-D activation request for each outgoing call
4	node(if-isdn)[if-name]# aoc-e explicit	Enables the reception of AOC-E messages and sends an AOC-E activation request for each outgoing call

In default, an ISDN network interface provides AOC information to the connected phones only if available, i.e. only if the call is routed to an ISDN user interface that is connected to a network providing AOC information. The following procedure enables the transmission of AOC message on an ISDN network interface even if

there is no AOC information from the network. In that case a message containing the value *noChargeAvailable* is sent.

Step	Command	Purpose
1	node(ctx-ip)[ctx-name]# interface isdn <if-name>	Go to the ISDN network interface, for which you want to enable AOC for all calls
2	node(if-isdn)[if-name]# aoc-s automatic	Enables the transmission of AOC-S messages even if there is no tariff information from the network for all calls
3	node(if-isdn)[if-name]# aoc-d automatic	Enables the transmission of AOC-D messages even if there is not charge information from the network for all calls
4	node(if-isdn)[if-name]# aoc-e automatic	Enables the transmission of AOC-E message even if there is no charge information from the network for all calls

The following procedure enables the transmission of AOC message on a per-call basis. That is AOC messages are sent by the connected phone only if configured for a per-call basis.

Step	Command	Purpose
1	node(ctx-ip)[ctx-name]# interface isdn <if-name>	Go to the ISDN network interface, for which you want to enable AOC on a per-call basis
2	node(if-isdn)[if-name]# aoc-s explicit	Enables the transmission of AOC-S messages even if there is no tariff information from the network on a per-call basis
3	node(if-isdn)[if-name]# aoc-d explicit	Enables the transmission of AOC-D messages even if there is not charge information from the network on a per-call basis
4	node(if-isdn)[if-name]# aoc-e explicit	Enables the transmission of AOC-E message even if there is no charge information from the network on a per-call basis

The following table shows an overview of the AOC variants:

	no aoc-x	aoc-x transparent	aoc-x automatic	aoc-x explicit
Default option	no	yes	no	no
ISDN User Interface (connected to a PBX switch etc.)				
No message from the network	No information forwarded to the peer interface	No information forwarded to the peer interface	No information forwarded to the peer interface	Sends an aoc-x request to the network. If the network rejects the request, no information is forwarded to the peer interface
AOC message from the network	No information forwarded to the peer interface	Information forwarded to the peer interface	Information forwarded to the peer interface	Information forwarded to the peer interface
ISDN Network Interface (connected to phones)				
Phone does not request AOC on a per-call basis	No information sent	Information sent as received from the network, no information sent if the network does not provide information	Always send information, <i>noChargeAvailable</i> sent if the network does not provide information	No information sent
Phone requests AOC on a per-call basis	No information sent	Information sent as received from the network, no information sent if the network does not provide information	Always send information, <i>noChargeAvailable</i> sent if the network does not provide information	Always send information, <i>noChargeAvailable</i> sent if the network does not provide information

ISDN DivertingLegInformation2 Facility

Trinity is now able to extract the redirecting information from the DivertingLegInformation2 Facility and to provide them to the call control. In the other direction, the redirecting information can be sent as DivertingLegInformation2 Facility in addition to the Redirecting Number Information Element.

Transmit direction

Mode: interface isdn <interface>

Step	Command	Purpose
1	<code>[name] (if-isdn)[interface]#[no] diversion emit</code>	Enables or disables transmitting of the DivertingLegInformation2 Facility.

*Receive direction***Mode:** interface isdn <interface>

Step	Command	Purpose
1	[name] (if-isdn)[interface]#[no] diversion accept	Enables or disables receiving of the Diverting-LegInformation2 Facility.

T1 Caller-Name Support

The ISDN implementation now supports reception and transmission of the caller-name on T1 links as it is used in NI2 networks according to Bellcore GR-1367-CORE. Transmission of the caller-name is part of the Calling Name Delivery (CNAM) service.

In previous build series (R3.20), the caller-name was already supported for DSS-1 networks using User-User information elements and for Q.SIG (PSS-1) networks using FACILITY messages. Now the caller-name is also supported for NI2 networks following the Bellcore standard.

As a prerequisite, the *caller-name* feature must be enabled on each ISDN interface in the CS context separately. This command now has additional arguments to configure the SETUP retention as follows:

In NI2 networks an incoming ISDN SETUP message may contain a *NameInformationFollowing* indication instead of the name. This means that the calling-party name is not available yet, but will be sent later, for example, after the dictionary database lookup in progress succeeded. If such an incoming ISDN call is internally routed to another network (e.g. to a SIP network or to a ISDN DSS-1 network), we must know the name before sending the initial INVITE or SETUP message towards the destination network. Therefore we must retain the SETUP message of the incoming ISDN call until the name is present. The caller-name command now allows you to configure the behavior of this SETUP retention mechanism. There are three possible options:

- **caller-name ignore-absence <timeout>:** This configuration command specifies the behavior for incoming ISDN calls. When a *NameInformationFollowing* indication is received with the SETUP message, the call-initiation is retained until the name is received or until this timeout elapses. After that, the call is forwarded to the configured destination interface. When forwarding a call without a caller-name to a SIP network, please note that there is no chance to send the caller-name later over SIP.
- **caller-name early-alerting <timeout>:** This configuration command specifies the behavior for incoming ISDN calls. Some networks only deliver the name after an alerting indication. These networks simulate the mid-ring name delivery feature of analog lines. If early alerting is enabled, we send back a faked ALERTING message after a configurable timeout when we receive a *NameInformationFollowing* indication. This command can be used together with the ignore-absence command. For example, you can configure an interface to first generate an ALERTING message and later forward the call anyway. If used that way, the early-alerting timeout should be smaller than the ignore-absence timeout.
- **caller-name send-information-following:** This configuration command specifies the behavior for outgoing ISDN calls. If there is no name from the originating network, the ISDN interface configured with this command sends a *NameInformationFollowing* indication to the remote side itself.

The following example enables and configures the caller-name feature on a T1 ISDN interface for incoming calls. If no name is present in the SETUP message, but the SETUP message contains the *NameInformationFollowing* indication, an ALERTING message is sent back after 500ms. If there is no name after additional 500ms the call is routed to the destination network anyway.

Mode: context cs / interface isdn

Step	Command	Purpose
1	node(if-isdn)#caller-name	Enables reception of the caller-name.
2	node(if-isdn)#caller-name early-alerting 500	(optional) If no name is present in an incoming ISDN call and if the incoming SETUP message contains the NameInformationFollowing indication, we send a fake ALERTING message after 500ms towards the caller. The SETUP message is retained for this period, i.e. the call is not forwarded to the configured destination. This step is optional. When not configured, an ALERTING message is faked after 2s by default. You can disable faking an ALERTING message by using the “no” form of the command. Note: If the ignore-absence timeout is also configured, the early-alerting timeout should have a smaller value than the ignore-absence timeout.
3	node(if-isdn)#caller-name ignore-absence 1000	(optional) If no name is present in an incoming ISDN call and if the incoming SETUP message contains the NameInformationFollowing indication, we forward the call to the routing destination anyway after 1000ms (500ms after faking the ALERTING message in this example). This step is optional. When not configured, the call is forwarded after 4s by default. You can disable forwarding a call without a name by using the “no” form of the command. Note: The specified timeout is measured starting at the reception of the SETUP message, not when the early-alerting timeout elapses.

The following example enables and configures the caller-name feature on a T1 ISDN interface for outgoing calls. It enables the transmission of the NameInformationFollowing indication (encapsulated into sent SETUP message) when no name is present from the originating network:

Mode: context cs / interface isdn

Step	Command	Purpose
1	node(if-isdn)#caller-name	Enables transmission of the caller-name.
2	node(if-isdn)#caller-name send-information-following	If no name has been received from the originating network a NameInformationFollowing indication is send encapsulated into the SETUP message for the outgoing ISDN call. This feature is disabled by default.

Caller-Name Facility Format on QSIG

The format of the calling name facility on QSIG can now be configured to achieve interoperability with the Siemens Hipath PBX. The default behavior is a local invoke identifier and the name argument is simple. To be compatible with Siemens Hipath, it can be changed to send a global invoke identifier and the name argument extended.

Mode: interface ISDN

Step	Command	Purpose
1	node(if-isdn)[name]#caller-name presentation [(simple extended)]	Sets the format for the caller name facility in QSIG. Default = <i>Simple</i>

Format Examples

An example of the simple format:

```
Facility : Invoke : invokeid: 00000001
global operation
invoke {
  present = 1
  local = 0
  argumentnamePresentationAllowedSimple = 'UserA'
}
```

An example of the extended format:

```
Facility : Invoke : invoked: 00000001
global operation
invoke {
  present = 1
  global = 1.3.12.9.0
  argumentnamePresentationAllowedExtended {
    nameData = 'UserA'
    characterSet = 1
  }
}
```

Tunneling of ISDN UUI1 Information Over SIP

Patton devices have the capability to translate User-to-User information from ISDN networks to SIP and vice-versa. This applies only for UUI1 with implicit activation. UUI1 with explicit activation, UUI2 and UUI3 are not supported. On ISDN the information is transported in a UserUser information element in the messages SETUP, ALERTING, CONNECT, DISCONNECT and RELEASE. The RELEASE message is used as transport only if it is the message which initiates the termination of the call.

The protocol on the SIP side uses a User-to-User header according to "draft-ietf-cuss-sip-uui-isdn-11 - Interworking ISDN Call Control User Information with SIP". A User-to-User header can be inserted in the initial INVITE, answers to INVITE, BYE and answers to BYE messages. In the case of interworking from ISDN to SIP the main part of the User-to-User header contains the protocol discriminator and the data from the ISDN UserUser information element encoded as base16. The protocol discriminator is not considered by the Device, it is transported transparently over SIP. The following header parameters with the exact values are appended to the header:

- encoding=hex

- purpose=isdn-uu
- content=isdn-uu

An example of such a User-to-User header when tunneling the IA5 character string "Hubert34" would look like:

```
User-to-User: 044875626572743334;encoding=hex;purpose=isdn-uu;content=isdn-uu
```

In the case of interworking from SIP to SIP the exact content and parameters of the incoming User-to-User header are forwarded.

Mode: interface isdn

Step	Command	Purpose
1	device(if-isdn)[<name>]# [no] user-to-user-information emit	Enables/Disables sending of UUI1 information over ISDN. Default is disabled.

Mode: interface sip

Step	Command	Purpose
1	device(if-sip)[<name>]# [no] user-to-user-information emit	Enables/Disables sending of UUI1 information over SIP. Default is disabled.

Configuring the Message Waiting Indication Feature for ISDN

Note Message Waiting Indication is programmed in three sections of Trinity, the FXS interface, ISDN interface, and the SIP Location service. The information below refers to information for configuring the Message Waiting Indication feature for ISDN. For information on configuring the Message Waiting Indication feature for SIP, see section “[Configuring the Message Waiting Indication Feature for SIP](#)” on page 556. For information on configuring the Message Waiting Indication feature for FXS, see section “[Configure Message-Waiting Indication \(MWI\) Feature for FXS](#)” on page 900.

The ISDN interface configuration mode enables notifications about new voice messages with a stuttered dial tone. With the stuttered dial tone, if there is a new message/messages in the voice mailbox, for 3 seconds after taking the phone off-hook the dial-tone will “stutter”, producing short delays between the dial-tone cadence.

You can enable the stutter dial tone feature with the **message-waiting-indication** command.

Step	Command	Purpose
1	[node](if-isdn)[name]#[no] message-waiting-indication stutter-dial-tone	Enables/Disables Message Waiting Indication through Stuttered Dial Tone

Configuring the Release Tone on ISDN DSS1 Network Interfaces

The following command enables/disables a release tone when sending a DISCONNECT message on the ISDN DSS1 network interface.

Mode: interface isdn

Step	Command	Purpose
1	[node](if-isdn)#[no] release-tone	Specifies if a release tone on disconnect should be played. Default: enabled

Configuring the ISDN User-side Timer T304

The timer T304 on the ISDN user-side is optional and can be disabled. The timer T304 on ISDN user-side is started when receiving a SETUP ACKNOWLEDGE message and entering the overlap sending state. Timer T304 is reset after each additional digit send and stopped when overlap sending is terminated by receiving a CALL PROCEEDING, ALERTING or CONNECT message. Timer T304 starts with 30 seconds and expiring causes a DISCONNECT to terminate the call.

Mode: cfg

Step	Command	Purpose
1	[node](cfg)[0/0]#[no] isdn t304-user	Configures the ISDN user-side timer. Default: enabled

Chapter 64 FXS Interface Configuration

Chapter contents

Introduction	825
Configuring Conference Routing	829
Configuring the Ringing-Cadence	830
Creating a new Ringing-Cadence Profile	831
Populating the Ringing-Cadence	831
Flushing the Ringing-Cadence	832
Displaying the Configuration of Ringing-Cadence Profiles	832
Using the Ringing-Cadence Profile on FXS interfaces	833
Configuring Distinctive-Ringing	833
Configuring the Message Waiting Indication Feature for FXS	835
Configuration	836
Configuring Supplementary Services	836
Configured Service Activation	837
Dynamic Service Activation/Deactivation/Interrogation	840
Service Invocation	841
Creating a new FXS Supplementary-Services Profile	841
Using the FXS Supplementary-Services Profile on FXS interfaces	841
Configuring Flash-Hook Processing	842
Call Hold	844
Subscription/Activation	844
Invocation (Mute)	845
Invocation (Additional Call)	846
Call Waiting	848
Subscription	848
Activation	849
Invocation	849
Call Waiting Reminder Ring	850
Subscription/Activation	851
Invocation	851
Drop Passive Call	852
Subscription/Activation	853
Invocation	853
Drop Active Call	855
Subscription/Activation	855
Invocation	855
Call Toggle	857
Subscription/Activation	857
Invocation	857
Call Transfer	859

Subscription/Activation	860
Invocation	860
Conference	862
Subscription/Activation	862
Invocation	864
Call Park	866
Subscription/Activation	866
Invocation	866
Advice-of-Charge (AOC) Tax-Pulses	868
Converting AOC Information into Tax-Pulses	868
Local Tax-Pulse Generation	870

Introduction

This chapter provides an overview of FXS interfaces and the tasks involved in their configuration. This chapter does not explain the basic configuration steps common to all CS interfaces. Information about basic interface configuration can be found in the general chapter about CS interfaces (see chapter 45, “CS Interface Configuration” on page 521).

An FXS interface represents a connection of an analog FXS port to the call control of Trinity. It provides the signaling of the exchange side of an analog line, allows incoming and outgoing calls on this line, controls the line events, tones and data path, and provides a set of supplementary services such as Call hold, Call Transfer, etc. There is a one-to-one relation between an FXS port and an FXS interface. As shown in figure 120, each FXS port must be bound to an interface to become functional. Please refer to chapter 68, “FXS Port Configuration” on page 885 to find out how to configure an FXS port and bind it to an FXS interface.

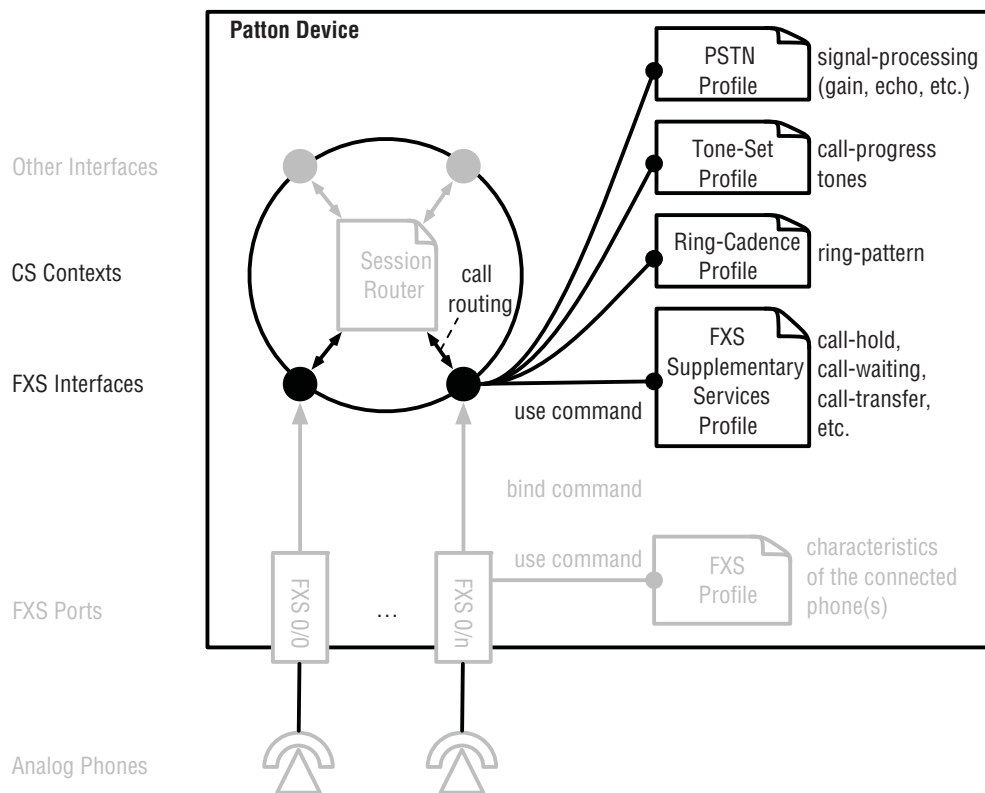


Figure 120. FXS-Interface configuration components

The table below provides an overview of all configuration commands available to configure the FXS interface:

Page	Command	Purpose
See section “Configuring Call Routing” on page 524	<code>node(ctx-cs)[ctx-name]# interface fxs if-name</code>	Enters FXS interface configuration mode. Creates a new interface if it does not exist yet.
	<code>node(if-fxs)[ctx-name.if-name]# [no] route call dest-interface if-name</code> or <code>node(if-fxs)[ctx-name.if-name]# [no] route call dest-table table-name</code> or <code>node(if-fxs)[ctx-name.if-name]# [no] route call dest-service service-name</code>	Specifies a destination interface for calls from the FXS phone (basic interface routing) or a destination table or call service (advanced call routing). Mandatory: Calls from the FXS phone are only processed if you configured a valid route destination. No destination is configured by default.
	7	<code>node(if-fxs)[ctx-name.if-name]# [no] route conference dest-service service-name</code>
See section “Configuring the Precall Service Tables” on page 528	<code>node(if-fxs)[ctx-name.if-name]# [reset] use mapping-table precall-service table-name</code>	Associates the precall service table table-name with the current FXS interface if-name. The precall service table is used to convert special numbers such as *61 into activation commands for supplementary services such as Call Waiting (see section “Configuring the Precall Service Tables” on page 528). Optional: Create and use a precall service table if you want the local subscriber to be able to activate/deactivate the Call-Waiting supplementary service. In this case you also have to enable Call Waiting in the fxs-supplementary-service profile.

Page	Command	Purpose
See section “Configuring the Interface Mapping Tables” on page 525	<code>node(if-fxs)[ctx-name.if-name]# [no] use mapping-table in [index] table-name</code> or <code>node(if-fxs)[ctx-name.if-name]# [no] use mapping-table out [index] table-name</code>	<p>Applies the specified list of mapping tables to all call properties. The referred mapping tables are not only applied during call setup but also during the call; this is different from mapping tables used in the Call Router. Interface mapping tables may be used to convert the connected-party number, for example.</p> <p>in: Converts call properties between the far-end party and the local FXS subscriber.</p> <p>out: Converts call properties between the local FXS subscriber and the far-end party.</p> <p>For example, if don't want the caller name of the far-end party to be sent to the subscriber's phone as part of the Caller-ID you should create a mapping table that deletes the name property and use this mapping-table for outbound conversions (out). As a consequence, this mapping is not only applied during the call-setup phase but also later during the call, e.g. for removing the connected-name property. (For more information, see section “Configuring the Interface Mapping Tables” on page 525).</p> <p>Optional: Create and use mapping-tables on the interface to convert numbering plans or to remove call-properties during all phases of a call.</p>
9	<code>node(if-fxs)[ctx-name.if-name]# [reset] use profile ringing-cadence profile-name</code>	<p>Associates the ringing-cadence profile <i>profile-name</i> with the current FXS interface <i>if-name</i>. The ringing-cadence profile defines the ring-pattern (ring and pause sequences) that is played towards the connected phone if a call is routed to this FXS interface.</p> <p>Optional: The default profile, <i>DEFAULT</i>, is used by default. It rings for 1 second, pauses for 4 seconds, then starts ringing again (standard European ring pattern).</p>
12	<code>node(if-fxs)[ctx-name.if-name]# [no] use profile ringing-cadence profile-name distinctive [index] (called-e164 calling-e164 called-alert-info) regex-pattern</code>	<p>Configures distinctive-ringing, i.e., overrides the ringing-cadence profile to <i>profile-name</i> for calls where the called-e164 number, the calling-e164 number, or the called-alert-info matches the specified regular expression pattern. This for example allows different ring patterns for different callers.</p> <p>Optional: No distinctive-ringing profiles are used by default.</p>

Page	Command	Purpose
See chapter 46, “Tone Configuration” on page 530	<code>node(if-fxs)[ctx-name.if-name]# [reset] use profile tone-set profile-name</code>	Associates the tone-set profile <i>profile-name</i> with the current FXS interface <i>if-name</i> . The tone-set profile maps call-progress tones such as dial-tone or ringback-tone to defined tone sequences (see chapter 46, “Tone Configuration” on page 530). Optional: The default profile, <i>DEFAULT</i> , is preconfigured with the Swiss standard call-progress tone mapping.
See chapter 43, “PSTN Profile Configuration” on page 495	<code>node(if-fxs)[ctx-name.if-name]# [reset] use profile pstn profile-name</code>	Associates the PSTN profile <i>profile-name</i> with the current FXS interface <i>if-name</i> . The PSTN profile contains the configuration for data/voice transmission on circuit-switched channels such as echo-canceller, input and output gain, etc. (see chapter 43, “PSTN Profile Configuration” on page 495). Optional: The default profile, <i>DEFAULT</i> , is preconfigured with typical neutral gain and echocanceler values.
13	<code>node(if-fxs)[ctx-name.if-name]# [reset] use profile fxs-supplementary-service profile-name</code>	Associates the FXS-supplementary-service profile <i>profile-name</i> with the current FXS interface <i>if-name</i> . That is, the supplementary services configured in the profile are enabled for the phone that is connected to the FXS port, which is bound to this FXS interface. Optional: The default profile, <i>DEFAULT</i> , is used by default. It has most of the available supplementary services enabled.

Example: Configuration of two FXS ports, each bound to an FXS interface, which routes the calls between the two ports (subscriber numbers 100 and 101, respectively).

```

profile tone-set DEFAULT
profile pstn DEFAULT
profile ringing-cadence DEFAULT
context cs SWITCH
  no shutdown

routing-table called-e164 RT-MAIN
  route 100 dest-interface IF-FXS-00
  route 101 dest-interface IF-FXS-01

interface fxs IF-FXS-00
  route call dest-table RT-MAIN

interface fxs IF-FXS-01
  route call dest-table RT-MAIN

port fxs 0 0
  subscriber-number 100

```



```

bind interface SWITCH IF-FXS-00
no shutdown

port fxs 0 1
subscriber-number 101
bind interface SWITCH IF-FXS-01
no shutdown

```

Configuring Conference Routing

The Patton device itself does not support conference calls. However, it is possible to use an external Nortel CS2K or RFC4240-compliant conference system to join calls into a conference using SIP. Such a setup involves three parties:

- 1 The local subscriber connected to an FXS port of the Patton device makes a conference call by following the steps below:
 - a Make or answer the first call
 - b Place the first call on hold
 - c Make or answer the second call
 - d Press the conference-invocation key sequence (default is Flash+3) to merge the two calls into a conference
- 2 The Patton device must make sure that a conference-invocation from the phone is forwarded to the conference server over SIP. This requires the administrator to change the configuration in different locations:
 - a Enable the conference supplementary service and configure which key sequence invokes the conference (see section “[Conference](#)” on page 862)
 - b Set up a conference-routing path from the FXS interface to the sip-conference service (section “[Configuring Conference Routing](#)”)
 - c Set up the sip-conference service in the Call Router (see section “[SIP Conference-service](#)” on page 772).
 - d Set up a call-routing path from the sip-conference service to a SIP interface (see chapter 49, “[Call Router Configuration](#)” on page 561).
- 3 The Nortel CS2K or RFC4240-compliant conference server must be set up to accept conference invocations from the Patton device.

Procedure: To route a conference invocation triggered by the phone to a conference service in the Call Router (step 2b).

Mode: Port FXS

Command	Purpose
<code>node(if-fxs)[ctx-name.if-name]# [no] route conference dest-service service-name</code>	Specifies a destination conference service that connects to an external conference server. Optional: No default conference destination is configured.

Example: Forward conference-invocation requests to a RFC4240-compliant soft switch.

This example shows how to enable the IP debug monitor and the debug output generated by the consecutive commands `ipaddress DHCP release` and `ipaddress DHCP renew` on IP interface *LAN*.

```
profile fxs-supplementary-services DEFAULT
  conference mode nortel pattern !3      # use Flash+3 to invoke

context cs SWITCH
  no shutdown

interface sip IF-SIP-CONFERENCE          # SIP interface to soft switch
  bind context sip-gateway SIP           # SIP gateway not shown here

service sip-conference SRV-CONFERENCE-RFC4240
  route call dest-interface IF-SIP-CONFERENCE
  conference-server 192.168.1.1         # IP address of soft switch

interface fxs IF-FXS-00
  route call ...                         # regular call routing
  route conference dest-service SVC-CONFERENCE-RFC4240
  use profile fxs-supplementary-services DEFAULT
```

Example: Forward conference-invocation requests from an FXS phone to a Nortel CS2K soft switch.

Note The user-part of the invocation URI must be changed to *conference*.

```
profile fxs-supplementary-services DEFAULT
  conference mode nortel pattern !3      # use Flash+3 to invoke

context cs SWITCH
  no shutdown

routing-table called-uri TBL-CONFERENCE
  route default dest-interface IF-SIP-CONFERENCE MAP-CONFERENCE

mapping-table called-uri to called-uri MAP-CONFERENCE
  map .%@(.) to sip:conference@\1      # replace user-part

interface sip IF-SIP-CONFERENCE          # SIP interface to CS2K
  bind context sip-gateway SIP           # SIP gateway not shown here

service sip-conference SRV-CONFERENCE-CS2K
  route call dest-table TBL-CONFERENCE  # route conference invocations
  conference-server 192.168.1.2         # IP address of CS2K server

interface fxs IF-FXS-00
  route call ...                         # regular call routing
  route conference dest-service SVC-CONFERENCE-CS2K
  use profile fxs-supplementary-services DEFAULT
```

Configuring the Ringing-Cadence

A ringing-cadence is a sequence of ring and ring-pause periods. The default ringing-cadence is configured to ring for one second, pause for four seconds, and repeat this sequence until the user picks up the receiver. This is

the standard European ring cadence. Because most likely the same ringing-cadences are used on multiple FXS interfaces the cadence itself is encapsulated into a ringing-cadence profile, which is used on the FXS interfaces.

Creating a new Ringing-Cadence Profile

Procedure: To create or enter a ringing-cadence profile.

Mode: Configure

Command	Purpose
<code>node(cfg) # [no] profile ringing-cadence profile-name</code>	Creates a ringing-cadence profile called <i>profile-name</i> if it does not exist yet and enters its configuration mode. There is a <i>DEFAULT</i> ringing-cadence profile (ring 1s, pause 4s), which can be modified. Since the <i>DEFAULT</i> profile is used on all FXS interfaces by default, changing the <i>DEFAULT</i> profile is the easiest way to change the ringing-cadence for all FXS interfaces.

Note If you delete a ringing-cadence profile that is used by FXS interfaces those interfaces will use the *DEFAULT* profile. The *DEFAULT* profile cannot be deleted.

Populating the Ringing-Cadence

Inside the ringing-cadence profile an arbitrary sequence of play/pause elements can be configured. The order in which the play/pause commands are entered (or appear in the running-config) defines the sequence in which the corresponding elements are played. The sequence will be repeated until the user picks up the receiver.

Procedure: To modify the ringing cadence within a profile.

Mode: Profile Ringing-Cadence

Command	Purpose
<code>node(pf-rc)[profile-name] # [no] (play pause) [index] duration</code>	If the <i>index</i> argument is missing, this commands appends a ring (play) or ring-pause (pause) period to the ringing-cadence. Otherwise, if an <i>index</i> of an existing element is given, that element is overwritten. A ringing-cadence should at least have a play and a pause element. The <i>duration</i> argument specifies how long the ring or ring-pause shall be applied in milliseconds. Enter a number within the range [100...60000], i.e., between 100ms and 60s.

Note If a ringing-cadence profile only contains one play or pause element, the element will be repeated forever, causing an endless ring or pause. An empty ringing-cadence will lead to an endless pause.

Example: Define a set of ringing cadence profiles for North America.

```

profile ringing-cadence US-CODE1
  play 1 2000
  pause 2 4000

profile ringing-cadence US-CODE2
  play 1 1000
  pause 2 1000
  play 3 1000
  pause 4 3000

profile ringing-cadence US-DISTINCTIVE
  play 1 500
  pause 2 500
  play 3 1000
  pause 4 1000
  play 5 500
  pause 6 2500

profile ringing-cadence US-SPECIAL
  play 1 500
  pause 2 500
  play 3 500
  pause 4 500
  play 5 1000
  pause 6 3000

```

Flushing the Ringing-Cadence

Inside the ringing-cadence profile an arbitrary sequence of play/pause elements can be configured. The order in which the play/pause commands are entered (or appear in the running-config) defines the sequence in which the corresponding elements are played. The sequence will be repeated until the user picks up the receiver.

Procedure: Delete all play/pause elements.

Mode: Profile Ringing-Cadence

Command	Purpose
<code>node(pf-rc)[profile-name] # flush-elements</code>	Resets the ringing-cadence, i.e., deletes all play and pause elements within the profile.

Displaying the Configuration of Ringing-Cadence Profiles

Procedure: Print the configuration of a ringing-cadence profile.

Mode: Operator Exec

Command	Purpose
<code>node> show profile ringing-cadence [name]</code>	Displays the configuration content of the ringing-cadence profile <i>name</i> . If the name argument is absent, displays a list of all ringing-cadence profiles.

Example:

```
node> show profile ringing-cadence
```

```

Ringing-Cadence Profiles:
=====

    DEFAULT
    US-CODE1
    US-CODE2
    US-DISTINCTIVE
    US-SPECIAL

node> show profile ringing-cadence US-SPECIAL

Ringing-Cadence Profile: US-SPECIAL
=====

Used: by                               4 module(s)

Ringing-cadence
-----
Play:                                   500ms
Pause:                                  500ms
Play:                                   500ms
Pause:                                  500ms
Play:                                   1000ms
Pause:                                  3000ms

```

Using the Ringing-Cadence Profile on FXS interfaces

Procedure: Use the ringing-cadence profile on an FXS interface.

Mode: Interface FXS

Command	Purpose
<code>node(if-fxs)[ctx-name.if-name]# [reset] use profile ringing-cadence profile-name</code>	Associates the ringing-cadence profile <i>profile-name</i> with the current FXS interface <i>if-name</i> . Optional: The default profile, <i>DEFAULT</i> , rings for 1 second, pauses for 4 seconds, then starts ringing again.

Configuring Distinctive-Ringing

Distinctive-Ringing lets the local subscriber identify a caller based on the ring pattern. The Patton device is able to play difference ring-patterns depending on the calling- or called-party number or explicitly through the SIP Alert-Info header. For example, different ring-patterns may be played for internal and external callers.

Distinctive ringing uses the same ringing-cadence profiles as described in the previous section. The only difference is how they are used in the FXS interface. For each interface the administrator may configure a list of (*match*, *profile*) pairs. The *match*-part defines in which circumstances the *profile* is used for ringing. The order of those list entries is relevant, since they are processed in the configured order. If an entry *matches*, the corresponding ringing cadence *profile* is used. If no entry matches, the ringing-cadence profile configured in the previous section is used (see section “Configuring the Ringing-Cadence” on page 830).

The *match*-part consists of a call-property type to examine and of a regular expression. You can use the following call-properties:

- **called-e164**: Choose a ring-cadence profile based on the called-party number
- **calling-e164**: Choose a ring-cadence profile based on the calling-party number
- **called-alert-info**: Choose a ring-cadence profile based on the content of the SIP Alert-Info header (without the angular brackets <>, e.g. use the *regex-pattern* http://127.0.0.1/Bellcore-dr5 to match the SIP Header Alert-Info: <http://127.0.0.1/Bellcore-dr5>)

Procedure: To add/remove distinctive-ringing rules to an FXS interface.

Mode: Interface FXS

Command	Purpose
<code>node(if-fxs)[ctx-name.if-name]# [no] use profile ringing-cadence profile-name distinctive [index] (called-e164 calling-e164 called-alert-info) regex-pattern</code>	Uses the ringing-cadence profile <i>profile-name</i> for calls where the called-e164 number, the calling-e164 number, or the SIP Alert-Info header matches the specified regular expression pattern. Optional: No distinctive-ringing profiles are used by default.

Example: Let the SIP switch define which of the standard Bell-Core ring-cadences to use.

```
profile ringing-cadence BELLCORE-RING-DEFAULT
  play 1 2000
  pause 2 4000

profile ringing-cadence BELLCORE-RING-2
  play 1 800
  pause 2 400
  play 3 800
  pause 4 4000

profile ringing-cadence BELLCORE-RING-3
  play 1 400
  pause 2 200
  play 3 400
  pause 4 200
  play 5 800
  pause 6 4000

profile ringing-cadence BELLCORE-RING-4
  play 1 300
  pause 2 200
  play 3 1000
  pause 4 200
  play 5 300
  pause 6 4000

profile ringing-cadence BELLCORE-RING-5
  play 1 500
  pause 2 60000
```

```

context cs SWITCH

interface fxs IF-FXS-00
use profile ringing-cadence BELLCORE-RING-DEFAULT
use profile ringing-cadence BELLCORE-RING-2 distinctive
                                1 called-alert-info http://127.0.0.1/Bell-
core-dr2
use profile ringing-cadence BELLCORE-RING-3 distinctive
                                2 called-alert-info http://127.0.0.1/Bell-
core-dr3
use profile ringing-cadence BELLCORE-RING-4 distinctive
                                3 called-alert-info http://127.0.0.1/Bell-
core-dr4
use profile ringing-cadence BELLCORE-RING-5 distinctive
                                4 called-alert-info http://127.0.0.1/Bell-
core-dr5

```

Configuring the Message Waiting Indication Feature for FXS

Note Message Waiting Indication is programmed in three sections of SmartWare, the FXS interface, ISDN interface, and the SIP Location service. The information below refers to information for configuring the Message Waiting Indication feature for FXS. For information on configuring the Message Waiting Indication feature for ISDN, see section “[Configuring the Message Waiting Indication Feature for ISDN](#)” on page 821. For SIP, see section “[Configuring the Message Waiting Indication Feature for SIP](#)” on page 556.

FXS interface configuration mode allows a selection of the way in which notifications about new voice messages are performed. Supported modes are:

1. Stuttered dial tone (if there is a new message/messages in users voice mailbox, for 3 seconds after taking the phone off-hook the dial-tone will “stutter”, producing short delays between the dial-tone cadence)
2. Visual Message Waiting Indication utilizing frequency shift keying signaling or line polarity reversal. Visual MWI-enabled FXS phones have a LED on the case to represent a new voice message. This LED can be turned on using one of the below configuration options.
 - Frequency shift keying signaling. The led will blink (or will simply light up) when a new message arrives to the voice mailbox.
 - Line polarity reversal (battery-reversal). The polarity is normal when no messages are waiting. When messages are waiting the polarity is reversed back and forth with an interval of 500ms, which makes the lamp on the phone blink with the same interval. As soon as the messages are cleared the battery-reversal stops and the lamp stops blinking. When the lamp on the phone is permanently on, but with no messages waiting, it could be an indication that the cable to connect the FXS port with the phone is crossed and should be replaced with a straight cable.
3. Stuttered dial tone and Visual Message Waiting Indication.

Configuration

You can enable both stutter dial tone and visual message waiting indication at the same time issuing both **message-waiting-indication** commands independently.

Mode: Interface FXS

Step	Command	Purpose
1	[node](if-fxs)[name]#[no] message-waiting-indication stutter-dial-tone	Enables/Disables Message Waiting Indication through Stuttered Dial Tone

Step	Command	Purpose
1	[node](if-fxs)[name]#[no] message-waiting-indication frequency-shift-keying	Enables Visual Message Waiting Indication through frequency shift keying signaling

Step	Command	Purpose
1	[node](if-fxs)[name]#[no] message-waiting-indication battery-reversal	Enables/ Disables using battery-reversal to display MWI on phones connected to FXS ports. Default is disabled.

Configuring Supplementary Services

Next to placing and receiving basic calls, FXS interfaces provide supplementary services to the local subscriber. These supplementary services are either provided locally by the Patton device or on a remote server. The following supplementary services are provided by the Patton device:

- Call Hold
- Call Waiting
- Call Waiting Reminder Ring
- Drop Passive Call
- Drop Active Call
- Call Toggle
- Call Transfer
- Conference
- Call Park
- AOC

These services are grouped into the following categories:

Service Category	Services	Description
Services to add an additional call	Call Hold Call Waiting	Allows the subscriber to make or accept two parallel calls. 2 nd outgoing call: After putting a call on hold the subscriber is able to make an additional consultation call. 2 nd incoming call: Allows an arriving call if the subscriber is already engaged in an active call.
Services that operate on two calls	Call Waiting Reminder Ring Drop Passive Call Drop Active Call Call Toggle Call Transfer Conference	If either the Call-Hold or the Call-Waiting service has been invoked, the subscriber is able to manage the two ongoing calls with these services.
Services that operate on a single call	Call Park	The subscriber is only able to park a call if there is one ongoing call.
Background services	AOC	Advice-of-Charge information is sent to the phone independently of the other services

Before a subscriber is able to *invoke* (trigger) a supplementary service, the service has to be *activated* (enabled). Activation must happen in the device configuration. However, the subscriber may deactivate, re-activate, or interrogate the state of a supplementary service through his phone.

Configured Service Activation

All services must be activated in the device configuration. This makes the services available to the local subscriber (i.e., to the connected phone). In Trinity, configured service activation is performed by enabling a service in an FXS supplementary-service profile that is used on the FXS interface. The syntax for those activation commands is discussed in the sections of the corresponding services below. The following table provides an overview of the available service-activation commands::

Page	Command	Purpose
18	<code>node(cfg) # [no] profile fxs-supplementary-services profile-name</code>	Enters FXS supplementary-services configuration mode. Creates a new profile if it does not exist yet.

Page	Command	Purpose
18	<code>node(pf-fxs-ss)[profile-name]# [reset] flash-hook (handle-locally relay)</code>	<p>Configures whether or not the flash-hook signal (on-hook for 80-750ms) is processed and how. Typically, the flash-hook signal is generated on the subscriber's phone by pressing the Flash key or by tapping the hook.</p> <p>handle-locally: The initial flash-hook signal is handled locally (default); it puts the ongoing call on hold and optionally allows the subscriber to make an additional consultation call (see section "Call Hold" on page 844).</p> <p>relay: The initial flash-hook signal is relayed to the far end, to e.g. a SIP switch.</p> <p>Optional: By default, the flash-hook is handled locally.</p>
18	<code>node(pf-fxs-ss)[profile-name]# [no reset] call-hold (mute additional-call)</code>	<p>Configures whether or not a call can be put on hold if the subscriber presses the Flash key. If enabled the command also configures whether or not the subscriber is able to make an additional consultation call.</p> <p>mute: The call is put on hold if the connected phone sends an initial flash-hook signal; both the local subscriber and the far-end party hears the hold-tone; the subscriber is not able to place a second call.</p> <p>additional-call: The call is put on hold if the connected phone sends an initial flash-hook signal; the far-end party hears the hold-tone; the subscriber hears the dial-tone and is able to make an additional consultation call (default).</p> <p>Optional: The call-hold service is enabled by default, and the subscriber is able to make an additional consultation call (additional-call).</p>
24	<code>node(pf-fxs-ss)[profile-name]# [no reset] call-waiting</code>	<p>Configures whether or not a second call can be accepted while the subscriber is involved in a first call. The subscriber hears the call-waiting tone and may invoke the Drop-Passive-Call (drops the waiting call), the Drop-Active-Call, or the Call-Toggle service to accept or reject the waiting call, or toggle between the calls, respectively. Those services must be enabled as well.</p> <p>Optional: The Call-Waiting service is enabled by default.</p>
27	<code>node(pf-fxs-ss)[profile-name]# [no reset] drop-passive-call [pattern pattern]</code>	<p>Configures whether or not the subscriber is able to drop the passive call (e.g. the waiting or held call). The <i>pattern</i> configures the keypad sequence the subscriber has to enter to activate the service; it usually starts with the '!' (Flash) key.</p> <p>Optional: The drop-passive-call service is enabled by default; the default invocation pattern is '!0' (Flash+0)</p>

Page	Command	Purpose
31	<code>node(pf-fxs-ss)[profile-name]# [no reset] drop-active-call [pattern pattern]</code>	<p>Configures whether or not the subscriber is able to drop the active and resume the passive call. The <i>pattern</i> configures the keypad sequence the subscriber has to enter to activate the service; it usually starts with the '!' (Flash) key.</p> <p>Optional: The drop-active-call service is enabled by default, the default invocation pattern is '!1' (Flash+1)</p>
33	<code>node(pf-fxs-ss)[profile-name]# [no reset] call-toggle [pattern pattern]</code>	<p>Configures whether or not the subscriber is able to toggle between the active and passive call. The <i>pattern</i> configures the keypad sequence the subscriber has to enter to activate the service; it usually starts with the '!' (Flash) key.</p> <p>Optional: The call-toggle service is enabled by default; the default invocation pattern is '!2' (Flash+2)</p>
35	<code>node(pf-fxs-ss)[profile-name]# [no reset] call-transfer</code>	<p>Configures whether or not the subscriber is able to connect the active to the passive call. No invocation pattern has to be configured; the service is invoked when the subscriber goes on-hook. Invocation is only possible if the second call was originated by the connected phone, i.e., you cannot transfer if the second call was incoming (Call Waiting), you must dial the second call.</p> <p>Optional: The call-transfer service is enabled by default.</p>
37	<code>node(pf-fxs-ss)[profile-name]# [no reset] conference [mode mode] [pattern pattern]</code>	<p>Configures whether or not the subscriber is able to connect itself, the active far-end, and the passive far-end to a conference with the help of an external conference server. The <i>mode</i> argument specifies the type of the external conference server; rfc4240 mode will be used if you enable the service without explicitly specifying the <i>mode</i> argument.</p> <p>rfc4240: Pushes the conference to a RFC4240-compliant SIP switch (default)</p> <p>nortel: Pushes the conference to a Nortel C2SK SIP switch</p> <p>The <i>pattern</i> configures the keypad sequence the subscriber has to enter to activate the service; it usually starts with the '!' (Flash) key. The '!3' (Flash+3) pattern will be used if you enable the service without explicitly specifying the <i>pattern</i> argument.</p> <p>Optional: The conference service is disabled by default.</p>

Page	Command	Purpose
40	<code>node(pf-fxs-ss)[profile-name]# [no reset] call-park [pattern pattern]</code>	Configures whether or not the subscriber is able to park a call to a specific number on an external park server. The pattern configures the keypad sequence the subscriber has to enter to activate the service. The service is invoked by first pressing the Flash key '!'. However, the pattern MUST omit this '!' prefix. Optional: The park service is enabled by default; the default invocation pattern is '*98'.
42	<code>node(pf-fxs-ss)[profile-name]# [no reset] aoc emit relay</code> <code>node(pf-fxs-ss)[profile-name]# [no reset] aoc emit generate amount (units tenths hundredths thousandths) (per-minute per-hour)</code> <code>node(pf-fxs-ss)[profile-name]# [reset] aoc emit units-per-pulse amount (units tenths hundredths thousandths)</code> <code>node(pf-fxs-ss)[profile-name]# [reset] aoc emit currency-per-pulse amount (units tenths hundredths thousandths)</code>	Configures whether or not to send tax-pulses to the connected phone. This Advice-of-Charge (AOC) information can be generated locally (generate) or passed transparently from the far-end (relay), or both at the same time. In both cases, the units-per-pulse and currency-per-pulse configuration specifies the charge unit or currency unit that corresponds to a sent tax pulse, respectively. units-per-pulse: Configures how many charging units have to be accumulated from the network until the Patton device sends another tax pulse to the FXS port. currency-per-pulse: Configures the amount of currency that has to be accumulated from the network until the Patton device sends another tax pulse to the FXS port. Optional: AOC relay is enabled whereas AOC local-generation is disabled by default; the default conversion is configured such that one tax-pulse corresponds to one charging unit or ten hundredths of a currency unit (e.g. 10 cents).

Dynamic Service Activation/Deactivation/Interrogation

Dynamic activation or deactivation of a service is performed by the subscriber itself by placing a dummy-call to a service-activation/deactivation number. Interrogation is used by the subscriber to check whether the service is enabled or disabled. For example, the subscriber is able to activate, deactivate, and interrogate the Call-Waiting service by dialing a configurable number pattern such as *43# to activate, #43# to deactivate, and *#43# to interrogate the Call-Waiting service.

Note Currently, only the Call-Waiting service can be activated dynamically, because this is the only supplementary service invoked by an incoming call and not by the subscriber.

Service Invocation

Invocation is the process of triggering the service, typically by pressing a keypad sequence on the subscriber's phone. Table 35 shows the default keypad sequences required to invoke the service provided by the Patton device locally.

Table 35. Default keypad sequences to invoke FXS supplementary services

Service	Default Keypad Sequence Pattern
Call Hold	Flash (!)
Call Waiting	(new incoming call during active call)
Call Waiting Reminder Ring	(on-hook)
Drop Passive Call	Flash+0
Drop Active Call (Accept Passive Call)	Flash+1
Call Toggle	Flash+2
Call Transfer	(on-hook)
Call Park	(Flash) (dial-tone) *98
Conference	Flash+3

Creating a new FXS Supplementary-Services Profile

Procedure: To create or enter an FXS supplementary-services profile.

Mode: Configure

Command	Purpose
<code>node(cfg) # [no] profile fxs-supplementary-services profile-name</code>	Associates the ringing-cadence profile <i>profile-name</i> with the current FXS interface <i>if-name</i> . Optional: The default profile, <i>DEFAULT</i> , rings for 1 second, pauses for 4 seconds, then starts ringing again.

Note If you delete a FXS supplementary-services profile that is used by FXS interfaces those interfaces will use the *DEFAULT* profile. The *DEFAULT* profile cannot be deleted.

Using the FXS Supplementary-Services Profile on FXS interfaces

Procedure: Use the FXS supplementary-services profile on an FXS interface.

Mode: Interface FXS

Command	Purpose
<code>node(if-fxs)[ctx-name.if-name]# [reset] use profile fxs-supplementary-service profile-name</code>	<p>Associates the FXS-supplementary-service profile <i>profile-name</i> with the current FXS interface <i>if-name</i>. That is, the supplementary services configured in the profile are enabled for the phone that is connected to the FXS port, which is bound to this FXS interface.</p> <p>Optional: The default profile, <i>DEFAULT</i>, is used by default. It has most of the available supplementary services enabled.</p>

Configuring Flash-Hook Processing

Analog phones are usually equipped with a ‘Flash’ button that is used to invoke supplementary services. Such a flash-hook signal is an on-hook condition of short duration. Patton devices recognize this signal if the on-hook duration lasts for 80 to 750 milliseconds. Trinity allows the administrator to configure whether the flash-hook signal is processed locally by muting the call or even by offering the subscriber a second dial-tone to place a consultation call, or whether the flash-hook signal is forwarded to the network, e.g. if a SIP PBX implements the supplementary services.

Procedure: Configure whether the flash-hook is processed locally or by the far-end PBX or soft switch.

Mode: Profile FXS Supplementary-Services

Command	Purpose
<code>node(pf-fxs-ss)[profile-name]# [reset] flash-hook (handle-locally relay)</code>	<p>Configures whether or not the flash-hook signal (on-hook for 80-750ms) is processed and how. Typically, the flash-hook signal is generated on the subscriber’s phone by pressing the Flash key or by tapping the hook.</p> <p>handle-locally: The initial flash-hook signal is handled locally (default); it puts the ongoing call on hold and optionally allows the subscriber to make an additional consultation call (see Call Hold service).</p> <p>relay: The initial flash-hook signal is relayed to the far end, to e.g. a SIP switch.</p> <p>Optional: By default, the flash-hook is handled locally.</p>

Note To relay the flash-hook signal to a SIP switch (if the call is routed to/from a SIP interface), you have to configure the flash-hook transmission method (DTMF, RTP, SIP signaling) on the VoIP profile. See section “[Configuring](#)

[DTMF Relay](#)” on page 476.

There are situations where the flash-hook signal is processed locally even though flash-hook relay is configured. Let us first examine the normal case when relaying flash-hook signals: As depicted in the lower part of [figure 121](#) on page 844, even if the subscriber presses the flash key (Call Hold), there is always only one call on Trinity’s FXS interface for that subscriber. The PBX may be configured to allow the subscriber to make a consultation call, but this second call is handled by the PBX (please consult the PBX manual for further information how to toggle or transfer calls on the PBX). An additional incoming call from that PBX should also be handled by the PBX (PBX Call Waiting).

However, if a call is routed from another network to the FXS interface while the subscriber is engaged in a call (local Call Waiting), the second call is offered to the subscriber if the Call Waiting service has been activated (upper part of [figure 121](#) on page 844). In this scenario, where a second call is waiting on the Patton device, the flash-hook signals are used to manage this local call even if flash-hook relay is configured. The flash-hook signals are relayed again as soon as there is only one call on the FXS interface.

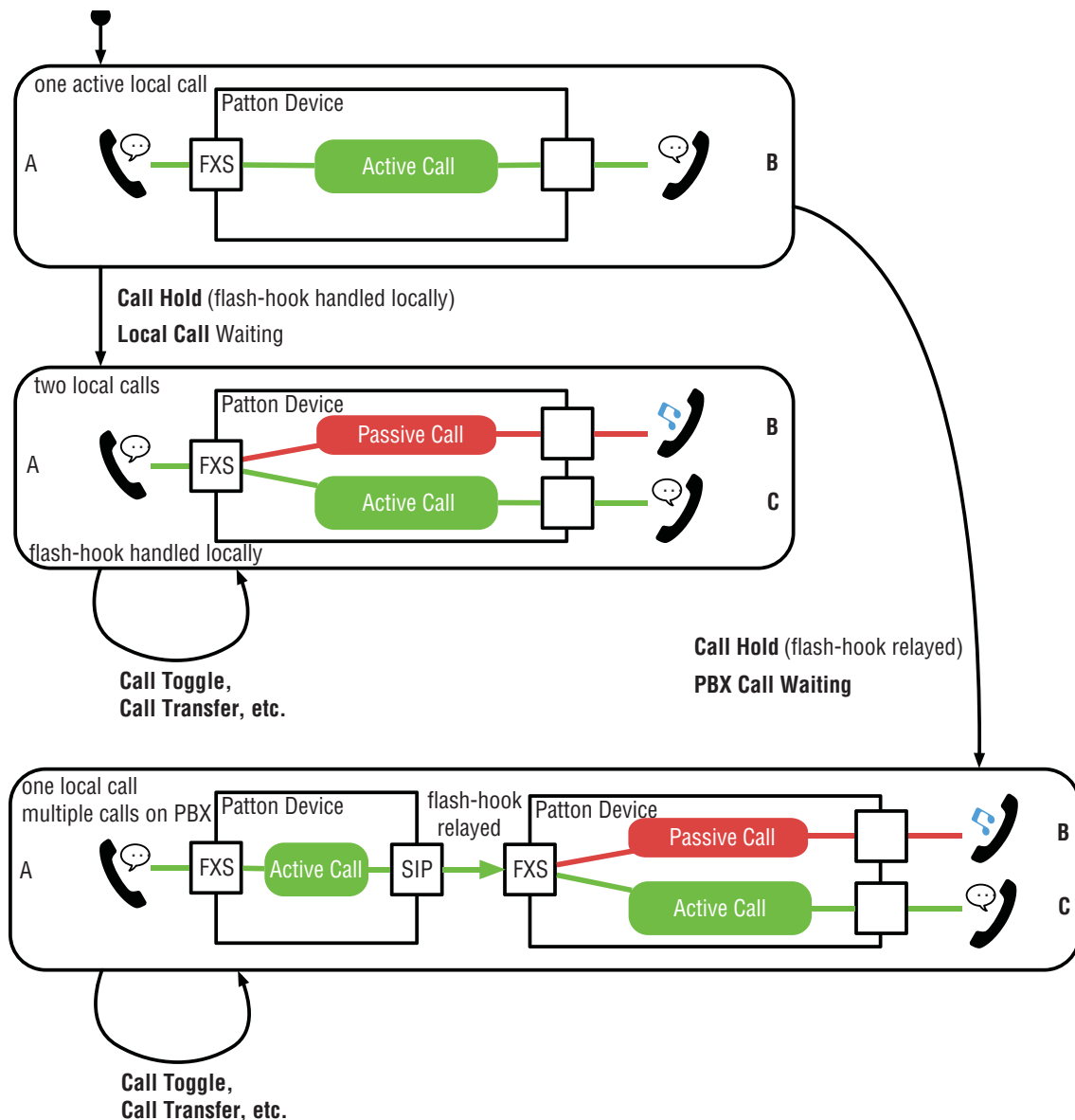


Figure 121. Local vs. Relayed Flash-Hook Processing

Call Hold

The Call-Hold service is used to set the far-end subscriber on hold. Depending on the configuration this just mutes the local subscriber or offers him/her the possibility to set up a second consultation call.

If enabled, the subscriber invokes the Call Hold service by sending the flash-hook signal. Depending on the configuration the flash-hook signal can be processed locally or forwarded to the far-end (e.g. to a SIP switch).

Subscription/Activation

The Call-Hold service can be activated in the FXS supplementary-services profile (it cannot be deactivated or re-activated by the subscriber through a service-activation keypad sequence).

To activate/deactivate Call Hold change the profile configuration bound to the subscriber's FXS interface as follows:

Procedure: Enable/Disable the Call-Hold supplementary service

Mode: Profile FXS Supplementary-Services

Command	Purpose
<code>node(pf-fxs-ss)[profile-name]# [no reset] call-hold (mute additional-call)</code>	<p>Configures whether or not a call can be put on hold if the subscriber presses the Flash key. If enabled the command also configures whether or not the subscriber is able to make an additional consultation call.</p> <p>mute: The call is put on hold if the connected phone sends an initial flash-hook signal; the far-end hears the hold-tone; the subscriber is not able to place a second call.</p> <p>additional-call: The call is put on hold if the connected phone sends an initial flash-hook signal; the far-end hears the hold-tone; the subscriber hears the dial-tone and is able to make an additional consultation call (default).</p> <p>Optional: The call-hold service is enabled by default, and the subscriber is able to make an additional consultation call (additional-call).</p>

Invocation (Mute)

Figure 122 on page 846 shows the call states and actions involved in invoking the Call Hold service if the following configuration has been applied:

- The flash-hook signal is handled locally
- The Call-Hold service has been activated with the **mute** option.

```
profile fxs-supplementary-services FXS-SS-HOLD-MUTE
  flash-hook handle-locally
  call-hold mute
```

Hold: If the local subscriber (A) is in an ongoing call with the far-end party (B), the subscriber invokes the Call-Hold service by pressing the Flash key. This puts the active call on hold. The local subscriber (A) will hear the hold tone. Whether B hears a hold tone, hold music, or another in-band message is under the responsibility of the far-end telephony system (PBX, soft switch, etc.).

Resume: If the subscriber (A) presses the Flash key again while the call to B is on hold, the call to B is resumed and A is able to talk to B again.

Drop: If the far-end party (B) drops the call while being on hold, the local subscriber (A) will hear the release tone. If A goes on-hook while the call is on hold, the call to B is dropped as well (not shown in the picture).

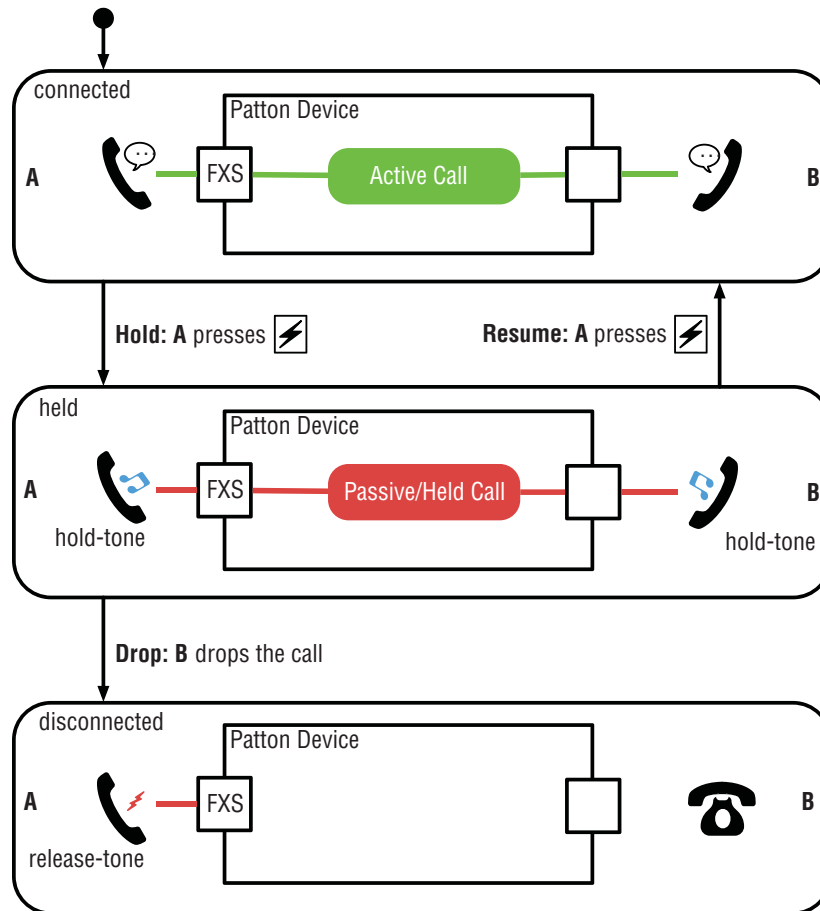


Figure 122. Call-Hold Supplementary-Service Invocation (no consultation call allowed)

Note Since the local subscriber is not allowed to place an additional call with the **mute** option of the Call-Hold command, it is not possible to reach a two-call state where other supplementary services such as Call Toggle, Call Transfer, etc. can be invoked. If this is desired, use the **additional-call** option instead (see section “[Invocation \(Additional Call\)](#)”).

Invocation (Additional Call)

Figure 123 on page 847 shows the call states and actions involved in invoking the Call Hold service if the following configuration has been applied:

- The flash-hook signal is handled locally
- The Call-Hold service has been activated with the **additional-call** option.

```
profile fxs-supplementary-services FXS-SS-HOLD-MUTE
  flash-hook handle-locally
  call-hold additional-call
```

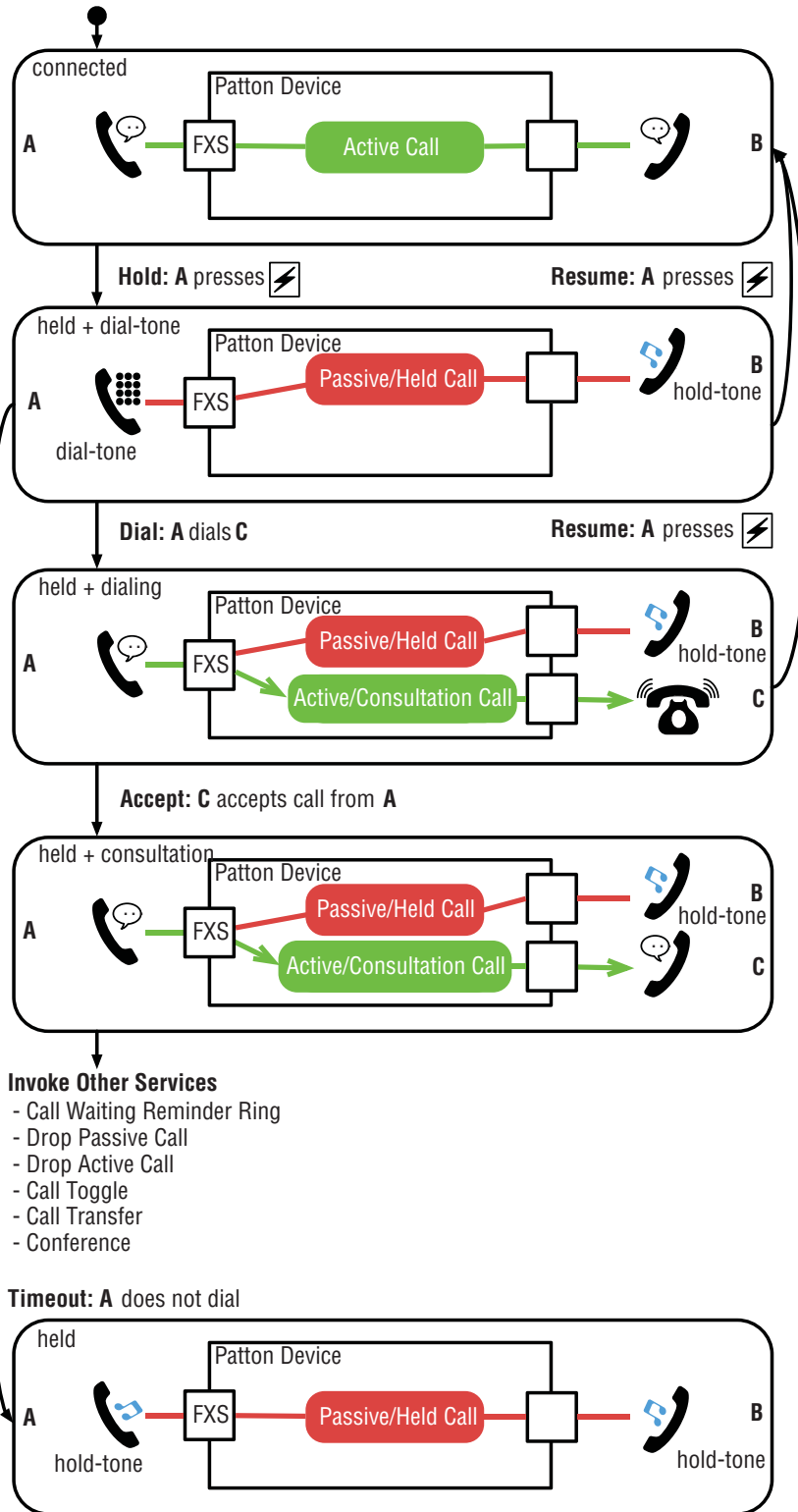


Figure 123. Call-Hold Supplementary-Service Invocation (additional consultation call allowed)

Hold: If the local subscriber (A) is in an ongoing call with the far-end party (B), the subscriber invokes the Call-Hold service by pressing the Flash key. This puts the active call on hold. The local subscriber (A) will hear the dial tone and is able to place an additional consultation call. Whether B hears a hold tone, hold music, or another in-band message is under the responsibility of the far-end telephony system (PBX, soft switch, etc.).

Consultation Call: If the subscriber (A) hears the dial tone, he/she is able to dial the number of third party (C) for consultation. Before C picks up the receiver the consultation call can be dropped and the held call resumed by pressing the Flash button again. After C picks up, toggling between the calls can be achieved with the pattern configured with the Call-Toggle service (see section “[Call Toggle](#)” on page 857).

Note The dial tone is only played for 12 seconds (or as configured with the address-completion timeout command in the CS context (see section “[Configure General Call Router Behavior](#)” on page 566). After this timeout the consultation call is closed and the hold tone is played.

Note Once the second call is connected, the supplementary services that involve two calls can be invoked. Please refer to the corresponding sections below:

- Call Waiting Reminder Ring
- Drop Passive Call
- Drop Active Call
- Call Toggle
- Call Transfer
- Conference

Call Waiting

The Call-Waiting service allows an additional call to be placed to a busy local subscriber. The subscriber can either accept or reject the call, or toggle between the active and the waiting call.

Subscription

The Call Waiting service can be activated in the FXS supplementary-services profile that is bound to the subscriber’s FXS interface:

Procedure: Enable/Disable the Call-Waiting supplementary service

Mode: Profile FXS Supplementary-Services

Command	Purpose
<code>node(pf-fxs-ss)[profile-name]# [no reset] call-waiting</code>	Configures whether or not a second call can be accepted while the subscriber is involved in a first call. The subscriber hears the call-waiting tone and may invoke the Drop-Passive-Call (drops the waiting call) or Drop-Active-Call or Call-Toggle service to accept or reject the waiting call, respectively. Those services must be enabled as well. Optional: The Call-Waiting service is enabled by default.

Activation

The Call-Waiting service is the only supplementary service that is not invoked by the subscriber but by the far-end party. This is why the subscriber should be able to deactivate the service him-/herself without changing the device configuration. For this purpose, the subscriber has to dial a special number.

However, you have to configure the Patton device to allow subscriber-based service activation. For this purpose, you have to use a precall-service mapping-table on the subscriber's FXS interface; the mapping-table defines which keypad sequence triggers Call Waiting activation, deactivation, and interrogation (see section "Configuring the Pre-call Service Tables" on page 528).

Example: Enables the Call Waiting service for the subscriber connected to FXS port 0/0. The subscriber is allowed to deactivate/reactivate/interrogate the service from his/her phone with the keypad sequences *43#, #43#, and *#43#, respectively.

```
profile fxs-supplementary-services FXS-SS-WAITING
  flash-hook handled-locally # don't relay the flash-hook
  call-waiting # enable Call Waiting

context cs SWITCH

precall-service-table PST-WAITING
  map *43# to activate-cw # activate Call Waiting
  map #43# to deactivate-cw # deactivate Call Waiting
  map *#43# to interrogate-cw # interrogate Call Waiting

interface fxs IF-PHONE
  use profile fxs-supplementary-service FXS-SS-WAITING
  use mapping-table precall-service PST-WAITING
```

- **Activation:** If the subscriber dials *43# the Call-Waiting service is (re-)activated. If the activation was successful the subscriber hears the confirmation tone, otherwise the release tone.
- **Deactivation:** If the subscriber dials #43# the Call-Waiting service is deactivated. If the deactivation was successful the subscriber hears the confirmation tone, otherwise the release tone.
- **Interrogation:** If the subscriber dials *#43# the Call-Waiting service is interrogated. If the service is active the subscriber hears the confirmation tone, if the service is inactive he/she hears the release tone.

Invocation

Figure 124 on page 850 shows the call states and actions involved in invoking the Call Waiting service if the following configuration has been applied:

- The Call-Waiting service has been activated.
- The subscriber has not deactivated the service.

```
profile fxs-supplementary-services FXS-SS-WAITING
  call-waiting
```

Note Note: If activated in the device configuration the Call-Waiting service is active even if the flash-hook signal is relayed to a remote PBX or soft switch.

If the local subscriber (A) is in an ongoing call with the far-end party (B), the Call-Waiting service is invoked by an additional call to the subscriber (A). The new call to C is still passive, C hears the ringback tone whereas the

waiting tone is played towards the subscriber (A). The subscriber can then use one of the supplementary services that involve two calls. Please refer to the corresponding sections below:

- Call Waiting Reminder Ring
- Drop Passive Call
- Drop Active Call
- Call Toggle
- Call Transfer
- Conference

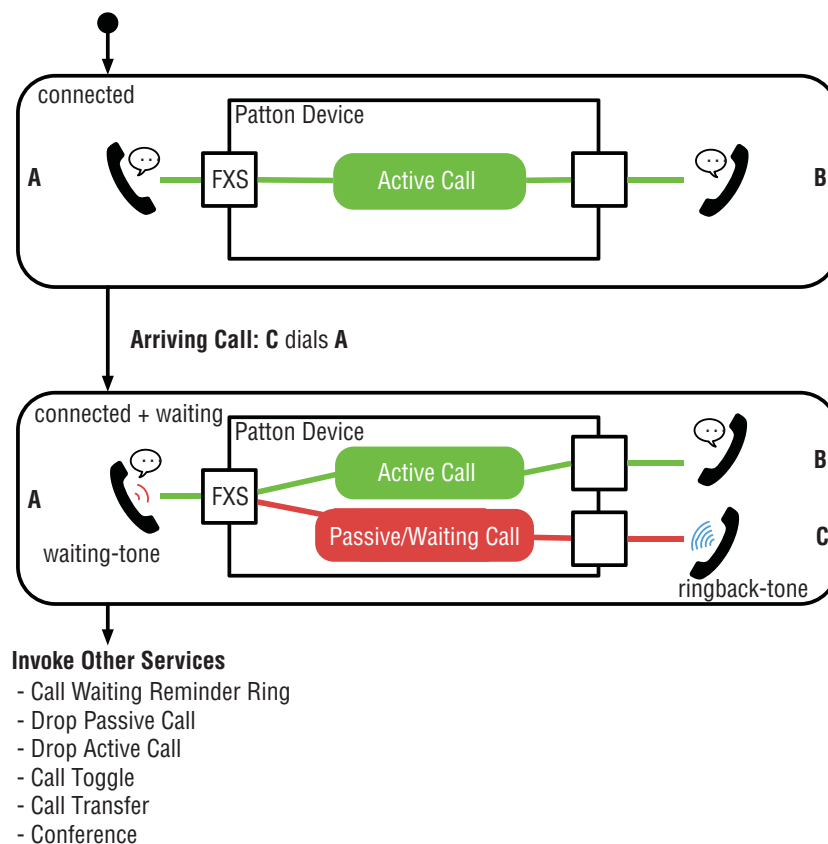


Figure 124. Call-Waiting Supplementary-Service Invocation

Call Waiting Reminder Ring

If the subscriber goes on-hook while there is an ongoing passive call, the phone rings. If the subscriber picks up the receiver, he/she is connected to the “lost” call, i.e., to the call that was passive before.

For example, the service may be invoked by the device in the following situations:

- **From Call Hold:** Putting a call on hold makes it passive (no voice is passed between the parties). If the Call-Hold service is enabled in the device configuration the subscriber is then able to make a new consultation call, which is the active call (see section “[Call Hold](#)” on page 844). If the Call-Transfer service is not enabled

and the subscriber goes on hook he/she terminates the consultation call; as a consequence, the phone starts to ring. When picking up the receiver he/she is **connected to the original call**. However, if the Call-Transfer service is enabled, going on-hook connects the original to the consultation call (see section “Call Transfer” on page 859).

- **From Call Waiting:** If a subscriber is engaged in an (active) call a new call may arrive. This new waiting call is the passive call in this scenario. If the subscriber goes on hook he/she terminates the first call; **the phone starts to ring for the waiting call**.

Subscription/Activation

The Call-Waiting-Reminder-Ring service is always active.

Invocation

Figure 125 on page 851 and figure 126 on page 852 show the call states and actions involved in invoking the Call-Waiting-Reminder-Ring service. The service is invoked by the Patton device automatically if the Call-Transfer service is disabled and the subscriber puts down the receiver while a passive call is still waiting.

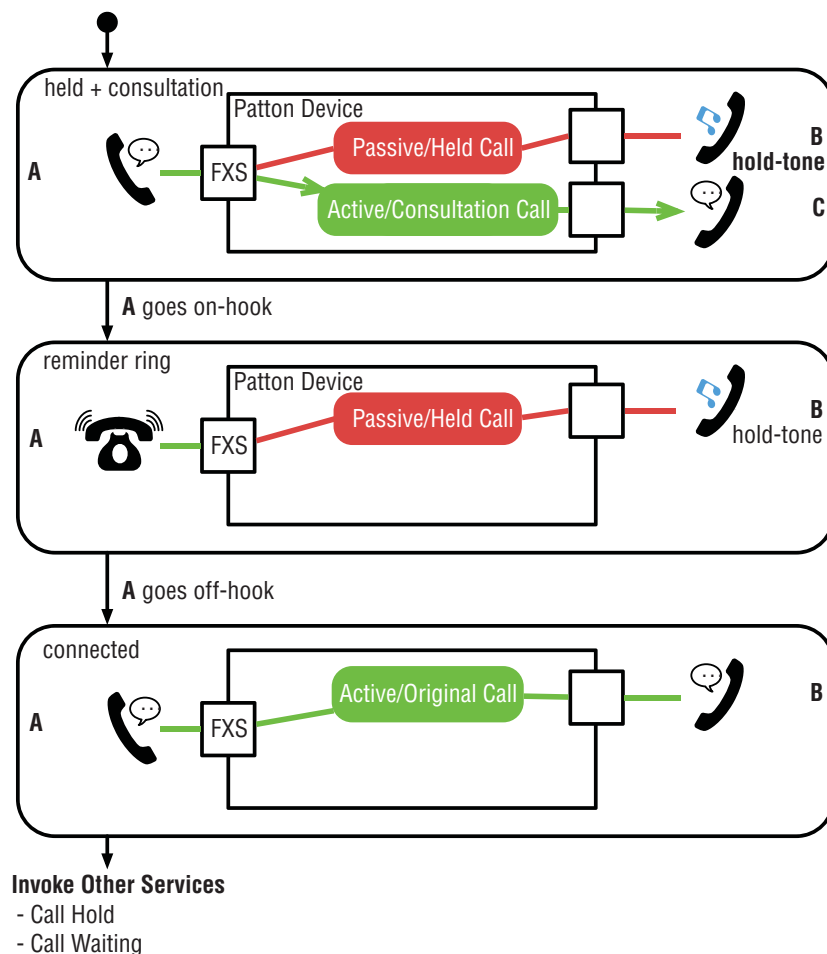


Figure 125. Call-Waiting-Reminder-Ring Supplementary-Service Invocation from Call Hold

Note If the Call-Transfer service is enabled when A puts down the receiver after having made a consultation call, B is transferred to C. Thus the Call-Transfer service, if enabled, has a higher priority (see section “Call Transfer” on page 859).

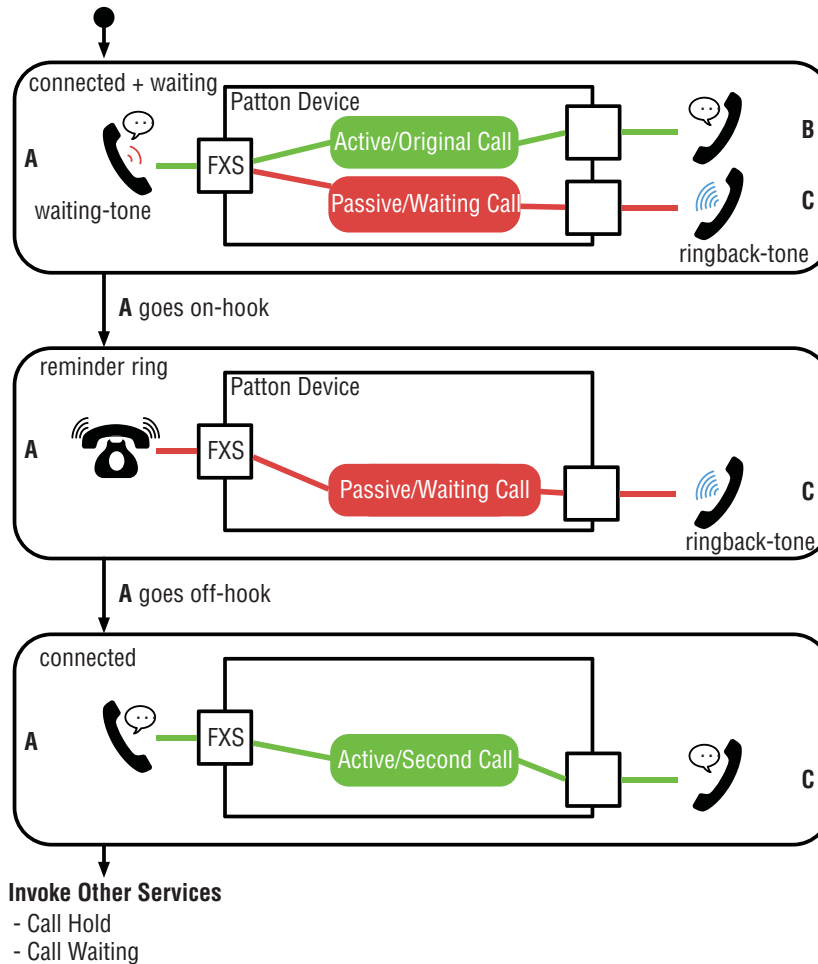


Figure 126. Call-Waiting-Reminder-Ring Supplementary-Service Invocation from Call Waiting

Note If A goes on hook with a waiting call, A's phone always rings independent whether the Call-Transfer service is enabled or not. This is because Call Transfer is not allowed to connect a waiting to an active call (see section “Call Transfer” on page 859).

Drop Passive Call

The Drop-Passive-Call service can be used if the local subscriber is engaged in two calls, one active and one passive call. It drops the passive call and stays connected with the active call.

This state is reached by either invoking the Call-Hold or the Call-Waiting service:

- **From Call-Hold:** Putting a call on hold makes it passive (no voice is passed between the parties). If allowed by configuration the subscriber is then able to make a new consultation call, which is the active call. In this scenario the Drop-Passive-Call service **drops the first call** and stays connected with the consultation call.
- **From Call-Waiting:** If a subscriber is engaged in an (active) call a new call may arrive. This new waiting call is the passive call in this scenario. By invoking the Drop-Passive-Call service the subscriber **rejects the waiting call**.

Subscription/Activation

The Drop-Passive-Call service can be activated in the FXS supplementary-services profile (it cannot be deactivated or re-activated by the subscriber through a service-activation keypad sequence). To activate/deactivate Drop Passive Call change the profile configuration bound to the subscriber's FXS interface as follows:

Procedure: Enable/Disable the Drop-Passive-Call supplementary service

Mode: Profile FXS Supplementary-Services

Command	Purpose
<code>node(pf-fxs-ss)[profile-name]# [no reset] drop-passive-call [pattern pattern]</code>	Configures whether or not the subscriber is able to drop the passive call (e.g. the waiting or held call). The <i>pattern</i> configures the keypad sequence the subscriber has to enter to activate the service; it usually starts with the '!' (Flash) key. Optional: The drop-passive-call service is enabled by default; the default invocation pattern is '!0' (Flash+0)

Invocation

Figure 127 on page 854 and figure 128 on page 854 show the call states and actions involved in invoking the Drop-Passive-Call service if the service has been activated. By pressing Flash+0 (or the pattern configured above) the passive call is dropped; the active call stays connected.

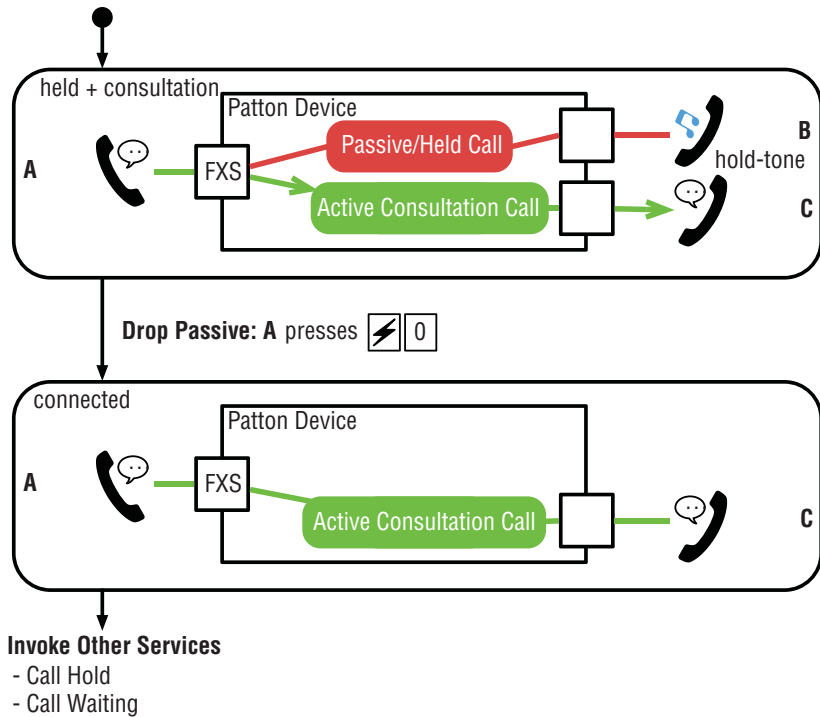


Figure 127. Drop-Passive-Call Supplementary-Service Invocation from Call Hold

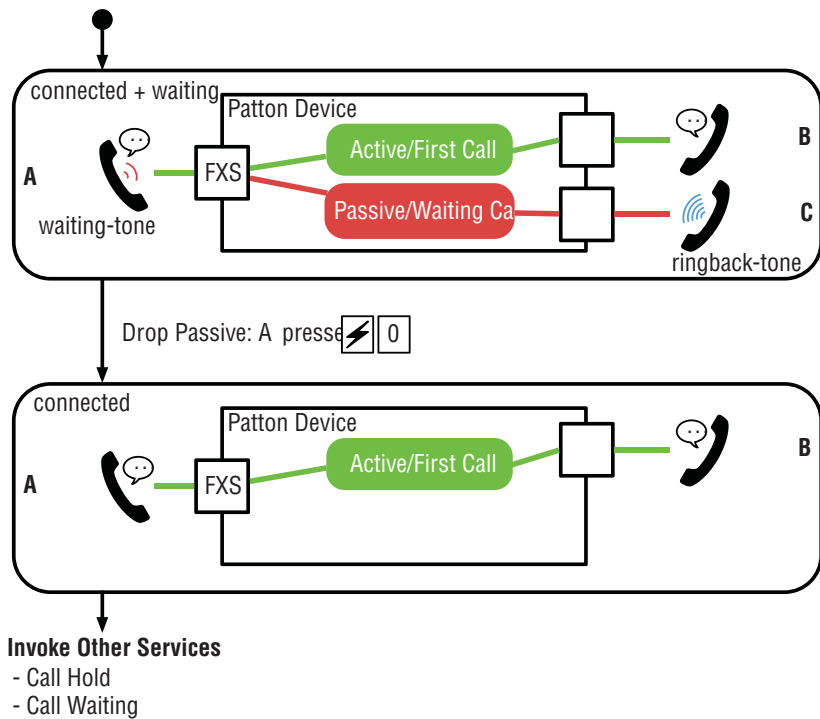


Figure 128. Drop-Passive-Call Supplementary-Service Invocation from Call Waiting

Drop Active Call

The Drop-Active-Call service can be used if the local subscriber is engaged in two calls, one active and one passive call. It drops the active call and activates the passive call.

This state is reached by either invoking the Call-Hold or the Call-Waiting service:

- **From Call-Hold:** Putting a call on hold makes it passive (no voice is passed between the parties). If allowed by configuration the subscriber is then able to make a new consultation call, which is the active call. In this scenario the Drop-Active-Call service **drops the consultation call** and returns to the first call.
- **From Call-Waiting:** If a subscriber is engaged in an (active) call a new call may arrive. This new waiting call is the passive call in this scenario. By invoking the Drop-Active-Call service the subscriber **drops the first call** and accepts the waiting call.

Subscription/Activation

The Drop-Active-Call service can be activated in the FXS supplementary-services profile (it cannot be deactivated or re-activated by the subscriber through a service-activation keypad sequence). To activate/deactivate Drop Active Call change the profile configuration bound to the subscriber's FXS interface as follows:

Procedure: Enable/Disable the Drop-Active-Call supplementary service

Mode: Profile FXS Supplementary-Services

Command	Purpose
<code>node(pf-fxs-ss)[profile-name]# [no reset] drop-active-call [pattern pattern]</code>	Configures whether or not the subscriber is able to drop the active and resume the passive call. The <i>pattern</i> configures the keypad sequence the subscriber has to enter to activate the service; it usually starts with the '!' (Flash) key. Optional: The drop-active-call service is enabled by default, the default invocation pattern is '!1' (Flash+1)

Invocation

Figure 129 on page 856 and figure 130 on page 856 show the call states and actions involved in invoking the Drop-Active-Call service if the service has been activated. By pressing Flash+1 (or the pattern configured above) the active call is dropped and the passive call is (re-)activated.

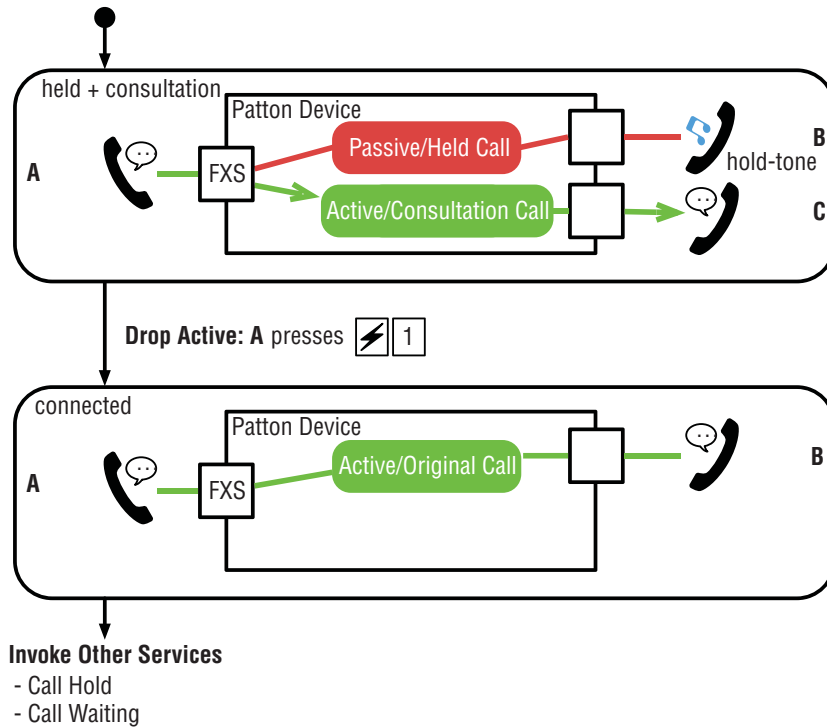


Figure 129. Drop-Active-Call Supplementary-Service Invocation from Call Hold

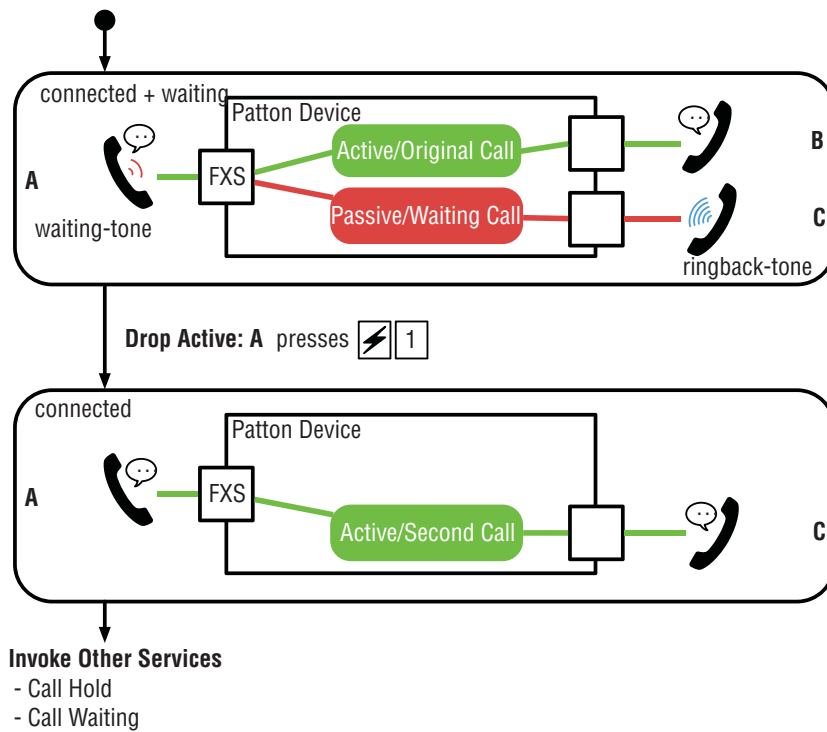


Figure 130. Drop-Active-Call Supplementary-Service Invocation from Call Waiting

Call Toggle

The Call-Toggle service can be used if the local subscriber is engaged in two calls, one active and one passive call. It toggles between the calls, i.e. it deactivates the active call and (re-) activates the passive call.

This state is reached by either invoking the Call-Hold or the Call-Waiting service:

- **From Call-Hold:** Putting a call on hold makes it passive (no voice is passed between the parties). If allowed by configuration the subscriber is then able to make a new consultation call, which is the active call. In this scenario the Call-Toggle service **returns to the first call and puts the consultation call on hold.**
- **From Call-Waiting:** If a subscriber is engaged in an (active) call a new call may arrive. This new waiting call is the passive call in this scenario. By invoking the Call-Toggle service the subscriber **puts the first call on hold and accepts the waiting call.**

Subscription/Activation

The Call-Toggle service can be activated in the FXS supplementary-services profile (it cannot be deactivated or re-activated by the subscriber through a service-activation keypad sequence). To activate/deactivate Call Toggle change the profile configuration bound to the subscriber's FXS interface as follows:

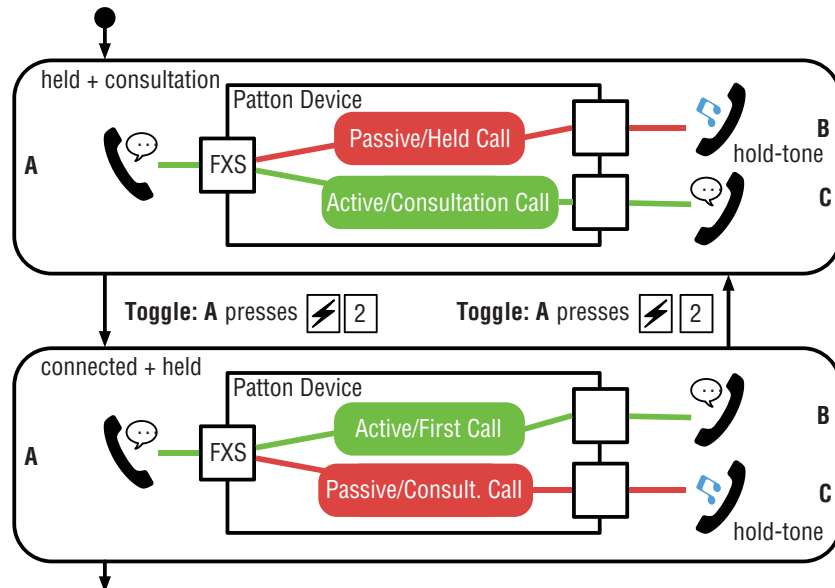
Procedure: Enable/Disable the Call-Toggle supplementary service

Mode: Profile FXS Supplementary-Services

Command	Purpose
<code>node(pf-fxs-ss)[profile-name]# [no reset] call-toggle [pattern pattern]</code>	Configures whether or not the subscriber is able to toggle between the active and passive call. The <i>pattern</i> configures the keypad sequence the subscriber has to enter to activate the service; it usually starts with the '!' (Flash) key. Optional: The call-toggle service is enabled by default; the default invocation pattern is '!2' (Flash+2)3

Invocation

Figure 131 on page 858 and figure 132 on page 859 show the call states and actions involved in invoking the Call-Toggle service if the service has been activated. By pressing Flash+2 (or the pattern configured above) the active call is deactivated and the passive call is (re-)activated.



Invoke Other Services

- Call Waiting Reminder Ring
- Drop Passive Call
- Drop Active Call
- Call Toggle
- Call Transfer
- Conference

Figure 131. Call-Toggle Supplementary-Service Invocation from Call Hold

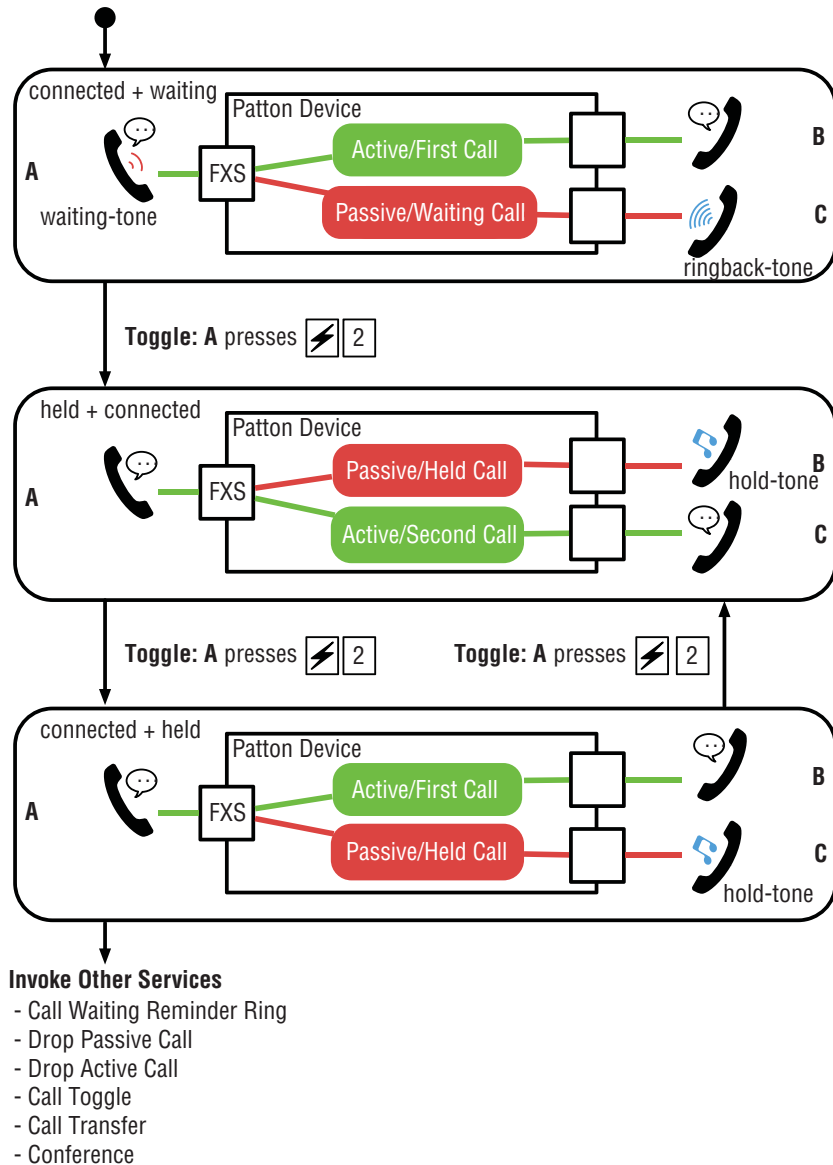


Figure 132. Call-Toggle Supplementary-Service Invocation from Call Waiting

Call Transfer

The Call-Transfer service can be used if the local subscriber is engaged in two calls, one passive/held and one active consultation call. The service cannot be invoked if the second call arrived with the Call-Waiting service. The Call-Transfer service is invoked if the subscriber goes on-hook, which causes the far-end parties of the held and the consultation calls to be connected together.

Subscription/Activation

The Call-Transfer service can be activated in the FXS supplementary-services profile (it cannot be deactivated or re-activated by the subscriber through a service-activation keypad sequence). To activate/deactivate Call Transfer change the profile configuration bound to the subscriber's FXS interface as follows:

Procedure: Enable/Disable the Call-Transfer supplementary service

Mode: Profile FXS Supplementary-Services

Command	Purpose
<code>node(pf-fxs-ss)[profile-name]# [no reset] call-transfer</code>	Configures whether or not the subscriber is able to connect the active to the passive call. No invocation pattern has to be configured; the service is invoked when the subscriber goes onhook. Invocation is only possible if the second call was originated by the connected phone, i.e., you cannot transfer if the second call was incoming (Call Waiting), you must dial the second call. Optional: The call-transfer service is enabled by default.

Note You also have to enable the Call-Hold service with the additional-call option. Otherwise the subscriber never reaches the state where he/she can transfer two calls. Refer to section “[Call Hold](#)” on page 844.

Invocation

[Figure 133](#) on page 861 shows the call states and actions involved in invoking the Call-Transfer service starting from a held and a consultation call. By going on-hook the subscriber transfers the two far-end parties together, i.e. connects the held call with the consultation call.

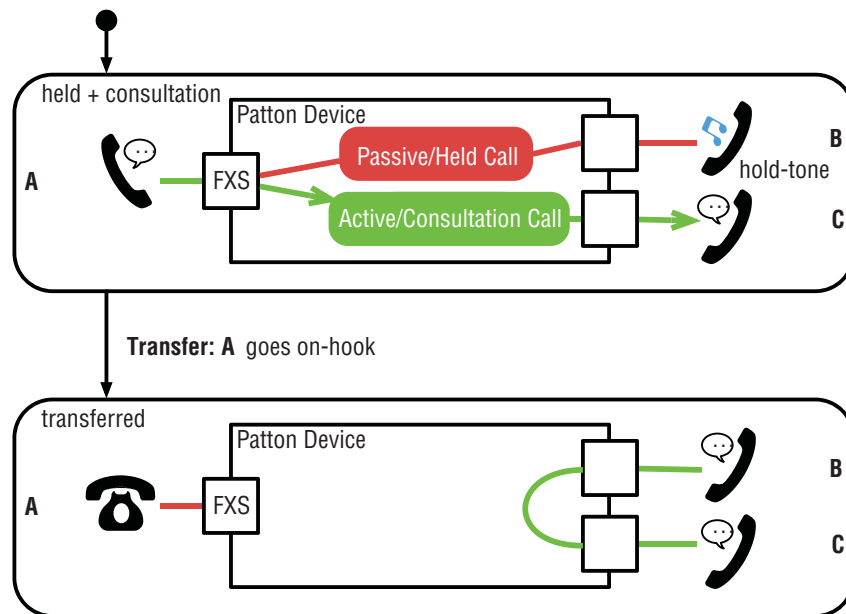


Figure 133. Call-Transfer Supplementary-Service Invocation from Call Hold

Figure 134 on page 862 shows the call states and actions involved when the subscriber tries to invoke the Call-Transfer service between a connected and a waiting call. This is not allowed since the subscriber could not yet verify the identity of the new caller. Imagine the subscriber is connected to his wife when his girlfriend calls in. If the subscriber goes off-hook he does not want his wife talking to his girlfriend.

Instead, if the subscriber goes on-hook in this scenario, it drops the active call and activates the Call-Waiting-Reminder-Ring service (see section “[Call Waiting Reminder Ring](#)” on page 850), i.e. the phone rings for the waiting call.

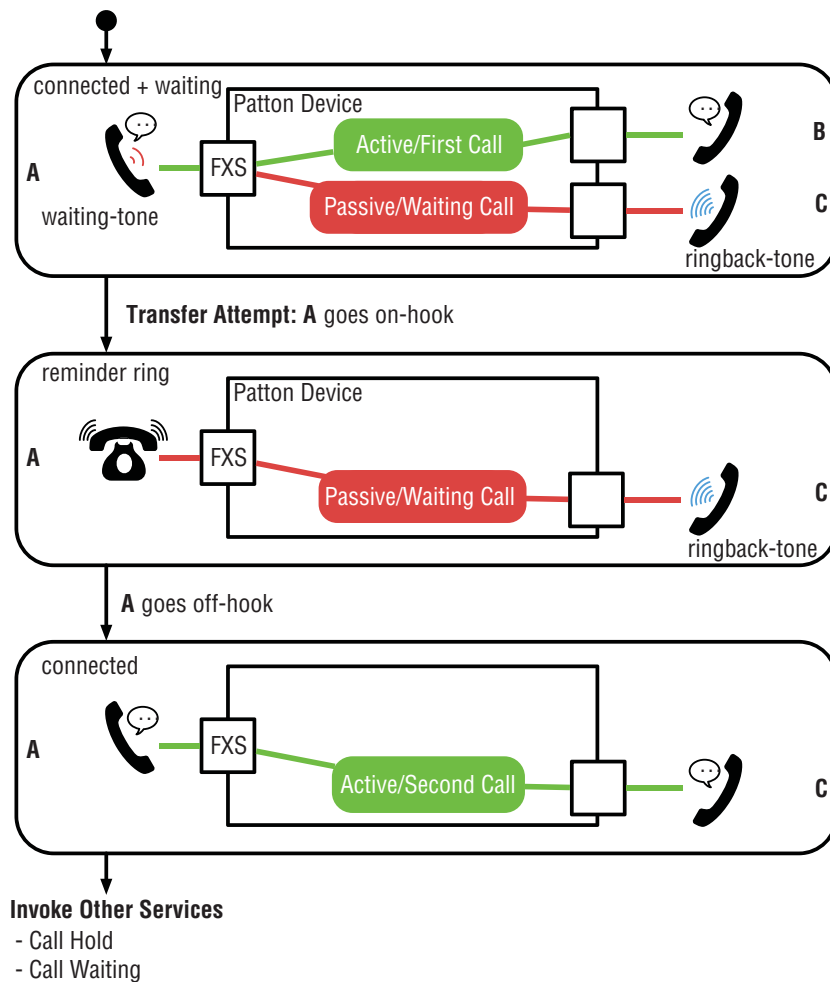


Figure 134. Call-Transfer attempt from Call Waiting; triggers Call Waiting Reminder Ring

Conference

The Conference service can be used if the local subscriber is engaged in two calls, one passive/held and one active consultation call. The service cannot be invoked if the second call arrived with the Call-Waiting service. The Patton device does not join the three call-legs together itself but forwards the conference to an external conference server (Nortel CS2K or RFC4240-compliant). Additional consultation calls can be added to an ongoing conference.

Subscription/Activation

The Conference service can be activated in the FXS supplementary-services profile (it cannot be deactivated or re-activated by the subscriber through a service-activation keypad sequence). To activate/deactivate Conference change the profile configuration bound to the subscriber's FXS interface as follows:

Procedure: Enable/Disable the Call-Transfer supplementary service

Mode: Profile FXS Supplementary-Services

Command	Purpose
<code>node(pf-fxs-ss)[profile-name]# [no reset] conference [mode mode] [pattern pattern]</code>	<p>Configures whether or not the subscriber is able to connect itself, the active far-end, and the passive far-end to a conference with the help of an external conference server. The mode argument specifies the type of the external conference server; rfc4240 mode will be used if you enable the service without explicitly specifying the mode argument.</p> <p>rfc4240: Pushes the conference to a RFC4240-compliant SIP switch (default)</p> <p>nortel: Pushes the conference to a Nortel C2SK SIP switch</p> <p>The <i>pattern</i> configures the keypad sequence the subscriber has to enter to activate the service; it usually starts with the '!' (Flash) key. The '!3' (Flash+3) pattern will be used if you enable the service without explicitly specifying the pattern argument.</p> <p>Optional: The conference service is disabled by default.</p>

Note In addition to activating the supplementary service you have to configure the following elements:

- The FXS interface that uses this profile has to forward invocation requests to a Conference Service in the Call Router (see section “[Configuring Conference Routing](#)” on page 829).
- You have to create a SIP Conference Service in the Call Router. The SIP Conference service forwards invocation requests to an external CS2K or RFC4240-compliant SIP Media Server (see section “[SIP Conference-service](#)” on page 772).

Example: Forward conference-invocation requests from a local FXS phone to a RFC4240-compliant soft switch.

```
profile fxs-supplementary-services DEFAULT
  conference mode nortel pattern !3      # use Flash+3 to invoke

context cs SWITCH
  no shutdown

interface sip IF-SIP-CONFERENCE        # SIP interface to soft switch
  bind context sip-gateway SIP         # SIP gateway not shown here
```

```

service sip-conference SRV-CONFERENCE-RFC4240
  route call dest-interface IF-SIP-CONFERENCE
  conference-server 192.168.1.1          # IP address of soft switch

interface fxs IF-FXS-00
  route call ...                          # regular call routing
  route conference dest-service SVC-CONFERENCE-RFC4240
  use profile fxs-supplementary-services DEFAULT

```

Example: Forward conference-invocation requests from a local FXS phone to a Nortel CS2K soft switch.

Note The user-part of the invocation URI must be changed to *conference*.

```

profile fxs-supplementary-services DEFAULT
  conference mode nortel pattern !3      # use Flash+3 to invoke

context cs SWITCH
  no shutdown

routing-table called-uri TBL-CONFERENCE
  route default dest-interface IF-SIP-CONFERENCE MAP-CONFERENCE

mapping-table called-uri to called-uri MAP-CONFERENCE
  map .%@(.) to sip:conference@1      # replace user-part

interface sip IF-SIP-CONFERENCE        # SIP interface to CS2K
  bind context sip-gateway SIP         # SIP gateway not shown here

service sip-conference SRV-CONFERENCE-CS2K
  route call dest-table TBL-CONFERENCE # route conference invocations
  conference-server 192.168.1.2       # IP address of CS2K server

interface fxs IF-FXS-00
  route call ...                       # regular call routing
  route conference dest-service SVC-CONFERENCE-CS2K
  use profile fxs-supplementary-services DEFAULT

```

Invocation

Figure 135 on page 865 shows the call states and actions involved in invoking the Call-Transfer service starting from a held and a consultation call. By pressing Flash+3 (or the pattern configured above) the subscriber initiates a conference between him-/her-self and the two far-end parties.

Once the conference is established, the initiator may press Flash again to make an additional consultation call (see “Call Hold” on page 844), i.e. to invite another person to the conference. Once the consultation call is established, the initiator presses Flash+3 (or the pattern configured above) again to add the new participant to the conference. The process may be repeated arbitrarily (please consult the manual of the conference server).

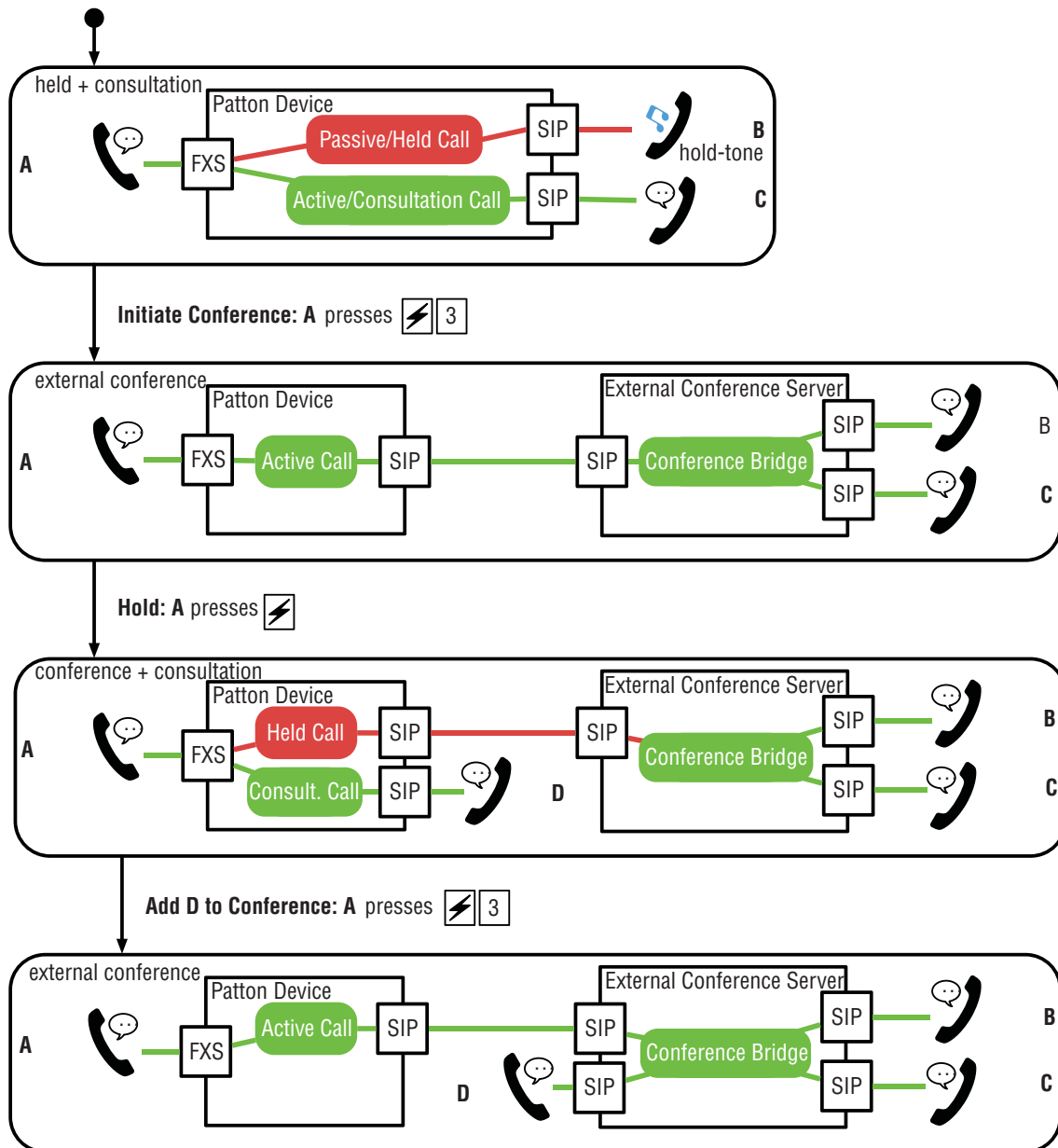


Figure 135. Conference Supplementary-Service Invocation from Call Hold

Note The Conference service can only be invoked if the second call was originated by the local subscriber (consultation call). If the second call is an incoming call (waiting call) and the local subscriber tries to invoke the Conference, the Call-Toggle service will be executed instead.

Note Once the conference is established the subscriber can only add new participants or hang up to leave the conference. Neither party is able to actively remove participants.

Call Park

The Call-Park service can be used to park a single ongoing call on a specific number. The Patton device does not park the call itself but blind-transfers the call to an external park server.

Subscription/Activation

The Call-Park service can be activated in the FXS supplementary-services profile (it cannot be deactivated or re-activated by the subscriber through a service-activation keypad sequence). To activate/deactivate Call Park change the profile configuration bound to the subscriber's FXS interface as follows:

Procedure: Enable/Disable the Call-Park supplementary service

Mode: Profile FXS Supplementary-Services

Command	Purpose
<code>node(pf-fxs-ss)[profile-name]# [no reset] call-park [pattern pattern]</code>	Configures whether or not the subscriber is able to park a call to a specific number on an external park server. The <i>pattern</i> configures the keypad sequence the subscriber has to enter to activate the service. Optional: The park service is enabled by default; the default invocation pattern is '*98'.

- Note** The Call-Park service is invoked by first pressing the Flash key (!) and waiting for the dial tone. However, the configured pattern **MUST** omit the '!' prefix, which is part of the Call-Hold service.
- Note** For the Call-Park service to work, you have to activate the Call-Hold service with the **additional-call** option, too.
- Note** In addition to activating the supplementary services you have to make sure that the far-end of the call (e.g. a SIP switch or PBX) understands a blind transfer to the configured service pattern.

Invocation

Figure 136 on page 867 shows the call states and actions involved in invoking the Call-Park service starting from an ongoing connected call. By pressing the Flash key, the call is put on hold and the subscriber hears the dial-tone. The subscriber then dials `*98parknumber#` (or instead of the prefix `*98` the pattern configured above). The park number is terminated by the pound key (#) or by waiting for 5 seconds after entering the last digit of the park number. The allocation of the park numbers is under the responsibility of the park server.

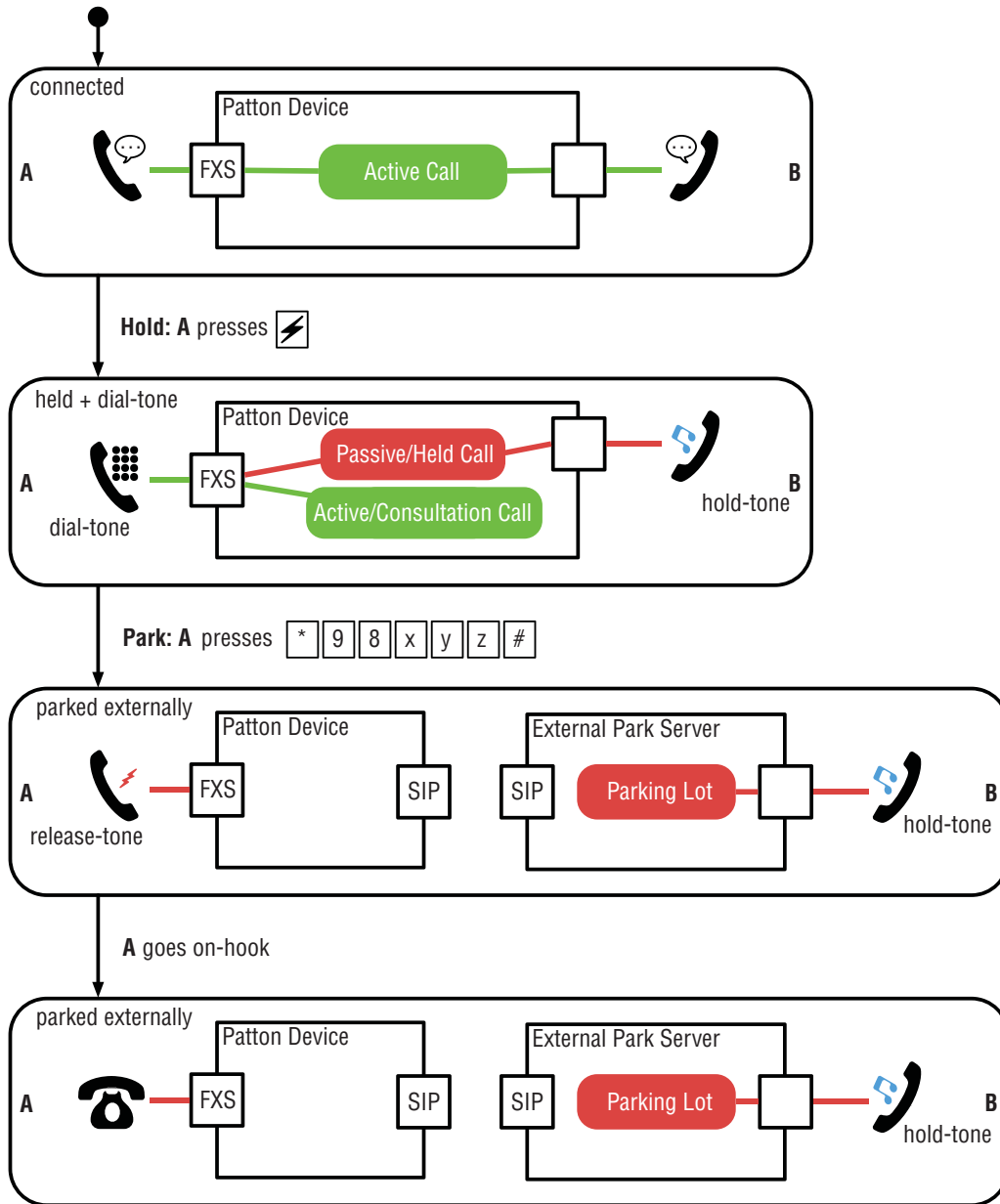


Figure 136. Call-Park Supplementary-Service Invocation

Note The Patton device sends a blind transfer to the park number. If the call is connected over SIP, the Patton devices sends a REFER message with the user-part of the Refer-To URI set to **98parknumber*.

Note: The process of picking up a call is defined by the park server. Typically, this is done by making a call to another service number (e.g. **99parknumber*). Please consult the user manual of the park server and make sure that the Call Router on the Patton device is configured to route that pick-up call correctly.

Advice-of-Charge (AOC) Tax-Pulses

AOC information received from ISDN or SIP can be forwarded to analog FXS lines by sending 12kHz or 16kHz tax pulses (see section “[Configure Tax-Pulse Modulation](#)” on page 899). The AOC information is converted from units or currency to a number of tax pulses which are emitted towards the connected phone or PBX.

AOC-D information received from ISDN or SIP during the call is converted into tax pulses. For subsequent AOC-D packets, the Trinity device computes the difference between the already sent tax pulses and only emits the tax pulses to reach the indicated total. For example: During a call, three AOC-D messages are received. The first results in 2 tax pulses and those pulses are emitted immediately. The second AOC-D message results in a total of 5 tax pulses; only the additional 3 pulses are emitted.

The same procedure is applied for AOC-E information, which is received from the ISDN or SIP side at the end of the call. However, we recommend that the ISDN/SIP network does not use AOC-E if the calls are forwarded to analog lines for the following reason: If the local FXS subscriber hangs up first, the Trinity device receives the AOC-E information later than that and cannot forward this information to the phone, because pulses can only be sent in off-hook state. We therefore recommend that the ISDN/SIP network sends charging updates as soon as they occur during the call by using AOC-D messages.

Converting AOC Information into Tax-Pulses

Advice-of-Charge (AOC) packets either contain charging units or a currency amount, depending on how the originating ISDN or SIP network is configured.

- **Charging Units:** The network sends charging units without currency information. Usually this reflects how many tax pulses the Patton device has to send to the phone, but there may be cases where the network assigns a charging unit a different currency amount than the phone does for a tax pulse (e.g. the network sends 5 units but we have to generate 10 pulses for for the phone to display the right charging information). This is why on the Patton device you are able to convert units into tax pulses.
- **Currency Amount:** The network sends charging information including the currency (e.g. USD 250 hundredths = USD 2.50). You are able to configure which currency amount shall be accumulated before the Patton device sends another tax pulse (e.g. one tax pulse for 10 hundredths = 10 cents).

[Figure 137](#) on page 869 shows how the Patton device converts AOC packets received from an ISDN or SIP network into tax pulses.

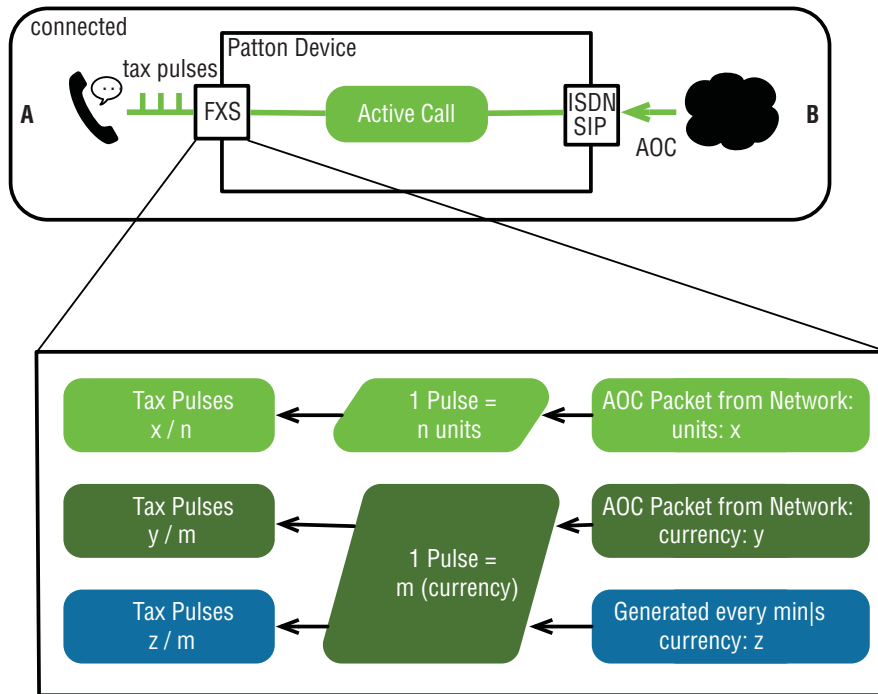


Figure 137. Advice-of-Charge to Tax-Pulse Conversion and Tax-Pulse Generation

There are two configuration commands that map AOC information into tax pulses, one to map charging units the other to map currency information. On an FXS interface, both types of information may arrive if the FXS interface terminates calls from different networks (e.g. from two different SIP switches).

Procedure: Configure how AOC charging and currency units are converted into tax pulses

Mode: Profile FXS Supplementary-Services

Command	Purpose
<pre>node(pf-fxs-ss)[profile-name]# [no reset] aoc emit relay</pre> <pre>node(pf-fxs-ss)[profile-name]# [reset] aoc emit units-per-pulse amount (units tenths hun- dredths thousandths)</pre> <pre>node(pf-fxs-ss)[profile-name]# [reset] aoc emit currency-per-pulse amount (units tenths hun- dredths thousandths)</pre>	<p>The relay command enables conversion of Advice-of-Charge (AOC) information from the ISDN or SIP network into tax pulses and emits them.</p> <p>The two other commands configure how many tax pulses the FXS port should emit when receiving an AOC packet from the network:</p> <ul style="list-style-type: none"> • units-per-pulse: Configures how many charging units have to be accumulated from the network until the Patton device sends another tax pulse to the FXS port. • currency-per-pulse: Configures the amount of currency that has to be accumulated from the network until the Patton device sends another tax pulse to the FXS port. <p>Optional: AOC relay is enabled by default; If not modified, one currency unit received from the network is mapped to one tax pulse (no conversion) and the currency amount of ten hundredths (e.g. 10 cents) is converted into one tax pulse.</p>

Note If the conversion results in a fractional number of tax pulses to be sent (e.g. 2.3) the Patton device rounds up the pulse count (e.g. 3 pulses are sent).

Example: The default AOC settings on the FXS supplementary-service profile.

```
profile fxs-supplementary-services DEFAULT
  aoc emit relay
  aoc emit units-per-pulse 1
  aoc emit currency-per-pulse 10 hundredths
```

Local Tax-Pulse Generation

In absence of Advice-of-Charge (AOC) information from the network the Patton device is able to generate tax pulses locally.

Procedure: Configure the profile to generate tax pulses

Mode: Profile FXS Supplementary-Services

Command	Purpose
<code>node(pf-fxs-ss)[profile-name]# [no reset] aoc emit generate amount (units tenths hun- dredths thousandths) (per-minute per-hour)</code>	<p>Instructs the FXS interface that uses this profile to generate tax pulses. The tax-pulse frequency or interval is not configured directly. You rather specify the currency you want to accumulate per minute or hour. This currency rate is then converted with the 'aoc emit currency-per-pulse' command (see above).</p> <p>Optional: AOC generation is disabled by default.</p>

Note The first tax pulse is sent immediately when the user picks up the phone.

Example: Ignore the AOC information received from the network and rather generate cost of 25 cents per minute; we assume that the phone treats one tax pulse as 10 cents.

```
profile fxs-supplementary-services DEFAULT
  no aoc emit relay                # don't relay AOC from net
  aoc emit currency-per-pulse 10 hundredths # phone: 1 pulse = 10 cents
  aoc emit generate 25 hundredths per-minute # cost: 25 cents / min
```

Chapter 65 **PRI Port Configuration**

Chapter contents

Introduction	873
Terminology	873
PRI Port Configuration task List	873
Enable/Disable PRI Port	873
Configuring PRI Port-type	874
Configuring PRI Clock-mode	874
Configuring PRI Line-code	874
Configuring PRI Framing	875
Configuring PRI Line-Build-Out (E1T1 in T1 mode only)	875
Configuring PRI Application Mode (E1T1 only)	875
Configuring PRI LOS Threshold (E1T1 only)	876
Configuring PRI Encapsulation	876
PRI Debugging	876

Introduction

This chapter provides an overview of the PRI (Primary Rate Interface) ports, their characteristics and the tasks involved in the configuration. The Patton devices know two different kinds of PRI ports, E1 and T1. This chapter describes the superset of all commands available on the different PRI ports. If a command is only executable for a specific port then this circumstance will be noted or highlighted in the command description. This chapter also explains how to prepare the ports for the usage of the different application protocols like ISDN or PPP. For some applications, there must be the possibility to access user defined sets of timeslots of an E1 or T1 port. On Patton devices' this feature is called a Channel Group and it will be described in this chapter as well.

Terminology

Hardware Type: Dependent on the device it can either be E1, T1 or E1T1. The Hardware Type and its belonging Slot and Port Number must be specified for entering the configuration mode of a port. It is not possible to change the Hardware Type, it is given by the system (i.e. BRI/PRI).

Port Type: This expression is used in relation with the E1T1 port and describes if the E1T1 port is currently running in E1 or in T1 mode. On an E1 or T1 port, the Port Type cannot be changed, it is static and matches the Hardware Type.

PRI Port Configuration task List

This section describes the configuration tasks for the PRI port.

- Enable/Disable PRI port (see page 873)
- Configuring the PRI port type (E1T1 only) (see page 874)
- Configuring PRI clock mode (see page 874)
- Configuring PRI line code (see page 874)
- Configuring PRI framing (E1T1 only) (see page 875)
- Configuring PRI line build out (E1T1 in T1 mode only) (see page 875)
- Configuring PRI application mode (E1T1 only) (see page 875)
- Configuring PRI LOS threshold (E1T1 only) (see page 876)
- Configuring PRI encapsulation (see page 876)
- PRI Debugging (see page 876)

Enable/Disable PRI Port

By default, the PRI port is disabled. The following command is used for enabling or disabling it.

Mode: port <hw-type> <slot> <port>

Step	Command	Purpose
1	<code>[name] (prt-e1t1)[slot/port]# [no] shutdown</code>	Enable/Disable the PRI port. Default: <i>shutdown</i> (which is disabled)

Configuring PRI Port-type

An E1T1 Port can either work in T1 or in E1 (G.704) mode. This mode can be changed dynamically as long as no encapsulation or encapsulation “ppp” is set. Be aware that changing the port-type also resets the framing and linecode parameters to the default values of the new port-type. If port-type change is not allowed due to current configuration, an error message will be issued.

Mode: port e1t1 <slot> <port>

Step	Command	Purpose
1	[name] (prt-e1t1)[slot/port]# port-type {e1 t1}	Changes operation mode of the port. Restriction: <i>Only available for e1t1 ports</i> Default: e1

Configuring PRI Clock-mode

The PRI Port can either work in clock-master or in clock-slave mode. This setting defines the clock dependency of the internal data processing. In clock-master mode the internal data processing is running on an independent clock source. In clock-slave mode the clock source for internal data processing is recovered from the receive line interface. Be aware that always a port-pair of clock-master and clock-slave are connected together. In the other case the data transmission will fail due to bit failures. This command also has the option ‘auto’ that can be used if the application running on the port is also of an asymmetric nature like master/slave, server/client or user/net. Normally, the option ‘auto’ is used if the port is setup for ISDN. In this case, the clock mode will automatically be derived from the Q.921 layer protocol. If the UNI-Side (User-Network Interface) of Q.921 is set to ‘net’, then the clock mode of the port is automatically set to ‘master’ and if Q.921 is configured as ‘user’ it will be set to ‘slave’.

Mode: port <hw-type> <slot> <port>

Step	Command	Purpose
1	[name] (prt-e1t1)[slot/port]# clock {auto master slave}	Configures the clock-mode of the port. Default: <i>master</i>

Configuring PRI Line-code

Two different line codes can be selected on the PRI port whereas only ‘ami’ is standardized for E1 and T1. If the port is running in E1 mode, ‘hdb3’ is also configurable and in T1 mode ‘b8zs’. If a linecode will be selected that is not standardized for the current port mode, an error message will be advised.

Mode: port <hw-type> <slot> <port>

Step	Command	Purpose
1	[name] (prt-e1t1)[slot/port]# linecode {ami b8zs hdb3}	Configures the line-code of the port. Default for e1: <i>hdb3</i> Default for t1: <i>b8zs</i>

Configuring PRI Framing

Three framing formats are available for selection on the E1/T1 port.

In structured mode, E1 can be configured for *crc4* or *non-crc4* and T1 has the framing option *esf* and *sf*.

- CRC4 (E1): Cyclic Redundancy Check 4. A CRC4 Multi-Frame consists of 16 continuous Basic-Frames. Each Multi-Frame can be divided into two Sub Multi-Frames. The first bit of Timeslot 0 of each even Sub Multi-Frame is called the C-Bit and belongs to the CRC4 check sum.
- ESF (T1): Extended Super Frame. The ESF is made up of 24 Basic-Frames. Each Basic-Frame includes one overhead bit, the F-Bit. The 24 F-Bits of one Extended Super Frame are used for synchronization (6 Bit), transmitting data link information (12 Bit) and for CRC6 calculation (6 Bit).
- SF (T1): Super Frame: The SF is made up of 12 Basic-Frames. Each Basic-Frame includes one overhead bit, the F-Bit. The 12 F-Bits of one Super-Frame represent the frame alignment pattern that is used for synchronization.

Mode: port <hw-type> <slot> <port>

Step	Command	Purpose
1	<code>[name] (prt-e1t1)[slot/port]# framing {crc4 non-crc4 esf sf}</code>	Configures the framing of the port. Restriction: <i>Only available for e1t1 ports</i> E1 mode formats are: <i>crc4, non-crc4, unframed</i> . T1 mode formats are: <i>esf or sf</i> . Default for e1: <i>crc4</i> Default for t1: <i>esf</i>

Configuring PRI Line-Build-Out (E1/T1 in T1 mode only)

The line build out configuration is used in long haul applications to prevent cross talk in the far end device.

Mode: port <hw-type> <slot> <port>

Step	Command	Purpose
1	<code>[name] (prt-e1t1)[slot/port]# line-build-out {0-133 133-266 266-399 399-533 533-655}</code>	Specifies the length attenuation in feet on the line interface. Restriction: <i>Only available for e1t1 ports in T1 mode</i> . Default for t1: <i>0-133 ft</i>

Configuring PRI Application Mode (E1/T1 only)

The PRI port can be configured to work in either short-haul or in long-haul mode. **Short-haul** is the default application and should be used for transmission distances up to **180m/600ft**. For transmission distances up to **1800m/6000ft**, select the **long-haul** application.

Mode: port <hw-type> <slot> <port>

Step	Command	Purpose
1	<code>[name] (prt-e1t1)[slot/port]# application {long-haul short-haul}</code>	Specifies the e1/t1 application mode Restriction: <i>Only available for e1t1 ports</i> Default: <i>short-haul</i>

Configuring PRI LOS Threshold (E1T1 only)

This command takes effect only if the PRI port is configured for **long-haul** applications. It specifies the sensitivity for **Loss Of Signal threshold**. A signal suffers more attenuation over long distances than over short distances. Therefore the LOS-Threshold must be set higher for longer transmission distances. This command has a default value of -46dB which should be enough for distances up to 1600 m/5250 ft.

Mode: port <hw-type> <slot> <port>

Step	Command	Purpose
1	<code>[name] (prt-e1t1)[slot/port]#los-threshold {-4dB -6dB -8dB ... -46dB -48dB}</code>	Specifies Loss Of Signal Threshold Restriction: <i>Only available for e1t1 ports</i> Default: <i>-46dB</i>

Configuring PRI Encapsulation

The PRI encapsulation command prepares the port for a specific application protocol.

The q921 encapsulation type automatically binds the signaling timeslot (D-channel) of the selected port to the ISDN Layer 2 protocol. This is timeslot 16 for an E1 and timeslot 24 for a T1 port. If in the q921 configuration mode q931 is specified as next encapsulation, the control of all remaining timeslots (B-channels) is given to the ISDN Layer 3 protocol. For more information please see Chapter 61, “[ISDN Overview](#)” on page 793 and Chapter 62, “[ISDN Configuration](#)” on page 798.

Mode: port <hw-type> <slot> <port>

Step	Command	Purpose
1	<code>[name] (prt-e1t1)[slot/port]#[no] encapsulation {q921}</code>	Specifies the encapsulation type of the PRI port. Default: <i>no encapsulation</i>

PRI Debugging

For the investigation of possible problems in link establishment, data transmission or synchronization, there exists a debug command with the options ‘event’ and ‘error’. The command has a hierarchical characteristic and can be applied to all ports of given type on the whole device, or to all ports of slot or just to one specific port.

Mode: Operator execution

Step	Command	Purpose
1	<code>[name]#[no] debug hw-type [([<slot> <port>])]</code>	Enables/Disables the PRI event/error monitor for the device a slot or a port. Examples: 1)[no] debug e1t1 Enables/Disables the event and the error monitor for all e1t1 ports of the device. -detail: detail level <0-3> (Def: 0) -full-detail: Enables maximum debug output

Mode: Operator execution

Step	Command	Purpose
1	<code>[name]#show port hw-type [[<slot> <port>]</code>	Prints information about the specified port with a given detail level.

Chapter 66 E1/T1 Port Configuration

Chapter contents

- Introduction879
- Patton support headquarters in the USA879
- Alternate Patton support for Europe, Middle East, and Africa (EMEA)879

Introduction

This chapter is incomplete. Please contact Patton Tech Support for assistance with configuring and troubleshooting E1/T1 ports.

Patton support headquarters in the USA

- Online support: Available at www.patton.com
- E-mail support: E-mail sent to support@patton.com will be answered within 1 business day
- Telephone support: Standard telephone support is available five days a week—from **8:00 am** to **5:00 pm EST (1300 to 2200 UTC/GMT)**—by calling **+1 (301) 975-1007**
- Support via VoIP: Contact Patton free of charge by using a VoIP ISP phone to call sip:support@patton.com
- Fax: **+1 (301) 869-9293**

Alternate Patton support for Europe, Middle East, and Africa (EMEA)

- Online support: Available at www.patton-inalp.com
- E-mail support: E-mail sent to support@patton-inalp.com will be answered within 1 business day
- Telephone support: Standard telephone support is available five days a week—from **8:00 am** to **5:00 pm CET (0900 to 1800 UTC/GMT)**—by calling **+41 (0)31 985 25 55**
- Fax: **+41 (0)31 985 25 26**

Chapter 67 **BRI Port Configuration**

Chapter contents

Introduction	881
BRI Port Configuration task List	881
Enable/Disable BRI Port	881
Configuring BRI Clock-mode	881
Configuring BRI Power-Feed	882
Configure BRI Line-Termination	882
Configuring BRI Encapsulation	882
BRI Debugging	882
BRI Configuration Examples	883
Example 1: ISDN with auto clock/uni-side settings	883
Example 2: ISDN with manual clock/uni-side settings	883

Introduction

This chapter provides an overview of the BRI (Basic Rate Interface) ports, their characteristics and the tasks involved in the configuration. A BRI port supports two 64kbit/s B-channels for switched voice or data connections, one 16kbit/s D-channel for signaling and always-on data transfer. This results a usable data bit rate of 144kBit/s.

BRI Port Configuration task List

This section describes the configuration tasks for the BRI port.

- Enable/Disable BRI port
- Configuring BRI clock mode
- Configuring BRI Power-Feed (see [page 882](#))
- Configuring BRI encapsulation (see [page 882](#))
- BRI Debugging (see [page 882](#))

Enable/Disable BRI Port

By default, the BRI port is disabled. The following command is used for enabling or disabling it.

Mode: port bri <slot> <port>

Step	Command	Purpose
1	<code>[name] (prt-bri)[slot/port]# [no] shutdown</code>	Enable/Disable the selected port. Default: <i>shutdown</i> (which is disabled)

Configuring BRI Clock-mode

The BRI Port can either work in clock-master or in clock-slave mode; the modes define the clock dependency of the internal data processing. In clock-master mode the internal data processing is running on an independent clock source. In clock-slave mode the clock source for internal data processing is recovered from the receive line interface. At all times, a port-pair of clock-master and clock-slave are connected together.

In the other case the data transmission will fail due to bit failures. The command below has the 'auto' option so it can be used if the application running on the port is also of an asymmetric nature like master/slave, server/client or user/net. Normally, the 'auto' option is used if the port is setup for ISDN. In this case, the clock mode will automatically derive from the Q.921 protocol. If the UNI-Side (User-Network Interface) of Q.921 is set to 'net', then clock mode of the port is automatically set to 'master' and in the other case to 'slave'.

Mode: port bri <slot> <port>

Step	Command	Purpose
1	<code>[name] (prt-bri)[slot/port]# clock {auto master slave}</code>	Configures the clock-mode of the port. Default: auto

Configuring BRI Power-Feed

Enable the application of power on the BRI port to provide power to ISDN terminals. This command applies only if the port is clock master (network side). It is only available on products with an internal, configurable ISDN power supply.

Mode: port bri <slot> <port>

Step	Command	Purpose
1	<code>[name] (prt-bri)[slot/port]#[no] power-feed</code>	Enables/Disables power-feed on the selected port. Default: disabled

Configure BRI Line-Termination

Switch internal termination impedance to on or off. Available for SN466x and SN467x devices for ports in TE/User/Slave mode.

Step	Command	Purpose
1	<code>[name] (prt-bri)[slot/port]#[no] line-termination</code>	Enables/Disables internal line-termination for TE ports. Default: disabled

Configuring BRI Encapsulation

The BRI encapsulation command prepares the port for a specific application protocol. After the right encapsulation type has been set, the configuration mode command for the selected protocol can be executed for protocol specific configuration.

q921: This encapsulation type automatically binds the signaling timeslot of the selected port to the ISDN Layer 2 protocol. For the BRI port this is the 16kbit/s D-channel. If in the q921 configuration mode q931 is specified as next encapsulation, the control of the two remaining timeslots (B-channels) is given to the ISDN Layer 3 protocol.

Mode: port bri <slot> <port>

Step	Command	Purpose
1	<code>[name] (prt-bri)[slot/port]#[no] encapsulation { channelized q921 }</code>	Specifies the encapsulation type of the BRI port. Default: q921

BRI Debugging

If a problem in the link is established, data transmission or synchronization, there are executable debug commands. The command has a hierarchical characteristic and can be applied to all ports on the whole device, or to all ports of a slot or just to one specific port. In addition, the 'show port' and 'show isdn' command can be used to print out information about the current configuration and about received and transmitted frames.

Mode: Operator execution

Step	Command	Purpose
1	<code>[node]#[no] debug bri { [<slot> [<port>]] [detail <level>] [full-detail] }</code> OR <code>[node]#[no] trace bri { [<slot> [<port>]] [alert critical emergency error info notice warning { debug [detail <level>]] [full-detail] } }</code>	Enables/Disables the numerous BRI traces viewed, and verbosity levels on a specific slot and BRI port. Default: no debug bri Examples: 1) [no] debug bri 0 0 full detail Enables/Disables all debugs on the BRI port 0 on slot 0 2) [no] trace bri 0 1 warning debug full detail Enables/Disables the warning trace in full detail on BRI port 1 on slot 0

Mode: Operator execution

Step	Command	Purpose
1	<code>[name]#show port bri [[<slot> <port>] [detail <level>]]</code>	Prints information about the specified port with a given detail level.
2	<code>[name]#show isdn bri [[<slot> <port>] [detail <level>]]</code>	Show the status and statistics of the specified ISDN port. The default detail level is 0.

BRI Configuration Examples

- Example 1: ISDN with auto clock/uni-side settings
- Example 2: ISDN with manual clock/uni-side settings

Example 1: ISDN with auto clock/uni-side settings

```
port bri 0 4
  power-feed
  encapsulation q921

q921
  uni-side auto

q931
  protocol dss1
  uni-side net
  bchan-number-order ascending
  bind interface bri04 switch

port bri 0 4
  no shutdown
```

Example 2: ISDN with manual clock/uni-side settings

```
port bri 0 4
  clock slave
  encapsulation q921
```

```
q921
  uni-side user

q931
  protocol dss1
  uni-side user
  bchan-number-order ascending
  bind interface bri04 switch

port bri 0 4 no shutdown
```


Chapter 68 **FXS Port Configuration**

Chapter contents

Introduction	886
Shutdown and Enable FXS Ports.....	888
Bind FXS Ports to FXS Interfaces in the CS Context.....	889
Configure a Subscriber Number	889
Configure Phone-Specific Parameters	890
Plan your FXS profiles	891
Use an existing FXS profile on an FXS port	894
Create, modify, reset, or delete an FXS profile	894
Configure the Electrical Standard	895
Configure the Caller-ID Presentation	896
Configure Connect and Disconnect Signals	897
Configure Tax-Pulse Modulation	899
Configure Message-Waiting Indication (MWI) Feature for FXS	900
Displaying the Configuration and State of an FXS port.....	901

Introduction

This chapter provides an overview of POTS signaling (Plain Old Telephone System) and describes the tasks involved in configuring FXS ports in Trinity.

This chapter includes the following sections:

- Shutdown and Enable FXS Ports (required)
- Bind FXS Ports to FXS Interfaces in the CS Context (required)
- Configure a Subscriber Number (recommended)
- Configure Phone-Specific Parameters (optional)
- Displaying the Configuration and State of an FXS port

FXS stands for *Foreign Exchange Station* and is the exchange side (or Central-Office (CO) side) of a POTS line. Even though POTS is seen as an old technology it still plays an important part in today's telecommunications networks. There are still a large number of analog phone sets in use worldwide and will do so in the future. These analog devices, be they phones, facsimile machines and the like, represent a large investment and it is desirable to have the technical means to hook such devices to a Voice over IP enabled network.

The signaling procedure used on FXS ports is different throughout different countries, but the basic idea is to use the POTS line itself as a current loop that signals off-hook and on-hook states of the handset. Going off-hook establishes a connection between the phone and whatever is on the other side (central-office switch, Patton device, etc.).

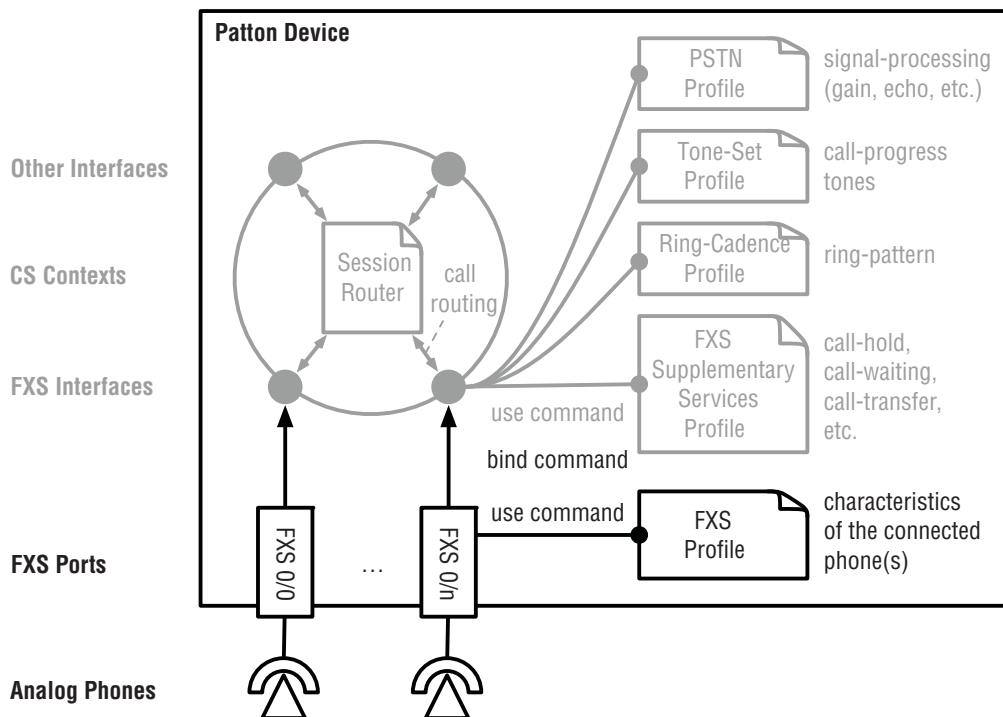


Figure 138. FXS-Port configuration components

Some Patton devices feature FXS ports, to which analog phones can be connected over RJ-11 connectors. In software physical ports are represented by enumerated port entries in the configuration, port fxs 0 0 ... port fxs 0 n. Each port uses an FXS profile, which defines the physical properties of the connected phone such as the electrical standard, whether or not the connected phone is able to indicate message-waiting state, and so on.

In order to receive and place calls, the port entry must be bound to an FXS interface in a CS context. The FXS interface contains all call-signaling related configuration such as the call-routing destination (outbound call interface) and the supplementary services offered to the phone (e.g. call-hold, call-waiting, call-transfer, etc.).

This chapter describes the configuration and operation principle of the FXS port. For a description of the FXS-interface configuration options, refer to chapter 64, “[FXS Interface Configuration](#)” on page 823. The table below provides an overview of all configuration commands available to configure the FXS port. A detailed step-by-step walk-through for all these configuration options can be found after the table and the example.

Page	Command	Purpose
	<code>node(cfg)# port fxs slot port</code>	Enters FXS port configuration mode. The port number matches the number on the label next to the port connector
6	<code>node(prt-fxs)[slot/port]# [no reset] subscriber-number number</code>	Configures the subscriber number for the current FXS port. The number argument is an arbitrary-length string consisting of the characters 0-9, *, or #. Recommended: Only if a subscriber number is configured the calling-party number will be announced for outgoing calls. No subscriber number is configured for the attached phone by default.
6	<code>node(prt-fxs)[slot/port]# [reset] use profile fxs name</code>	Associates the FXS profile name with the current FXS port <i>slot/port</i> . That is, the phone-characteristics configured in the profile are applied to this FXS port. Optional: The default profile, <i>DEFAULT_EU</i> , works well for most European phones (see table 36 on page 891). Use the <i>DEFAULT_US</i> profile instead for Northern American phones or create a new profile. Also consider fine-tuning the profile to enable additional features such as MWI, tax pulses, etc.
5	<code>node(prt-fxs)[slot/port]# [no] bind interface [context] interface</code>	Binds the FXS port to an <i>interface</i> in a CS context. If the <i>context</i> argument is omitted, the port is bound to the specified <i>interface</i> in CS context <i>SWITCH</i> . Mandatory: You must bind the port in order to place and receive calls. By default, the port is not bound to an interface.
4	<code>node(prt-fxs)[slot/port]# [no] shutdown</code>	Shuts down or activates the port. Mandatory: You must activate the port in order to place and receive calls. By default, the port is shut down.

Example: Configuration of two FXS ports, both using the default FXS profile for Northern America. Each FXS port is bound to an FXS interface, which routes calls to the other port (subscriber numbers 100 and 101, respectively).

```

profile fxs DEFAULT_US
  electrical-standard bell # default
  caller-id standard fsk-bell presentation mid-ring # default

context cs SWITCH
  no shutdown

routing-table called-e164 RT-MAIN
  route 100 dest-interface IF-FXS-00
  route 101 dest-interface IF-FXS-01

interface fxs IF-FXS-00
  route call dest-table RT-MAIN

interface fxs IF-FXS-01
  route call dest-table RT-MAIN

port fxs 0 0
  subscriber-number 100
  use profile fxs DEFAULT_US
  bind interface SWITCH IF-FXS-00
  no shutdown

port fxs 0 1
  subscriber-number 101
  use profile fxs DEFAULT_US
  bind interface SWITCH IF-FXS-01
  no shutdown

```

Note In the running-config all FXS profiles always print the most important configuration commands, i.e. the *electrical-standard* and *caller-id* commands, even if they reflect the default values.

Shutdown and Enable FXS Ports

Like all other ports FXS ports are disabled by default. However, your device may be shipped with a configuration that enables the FXS ports. Make sure to enable the ports before using them.

Procedure: To enable or shutdown an FXS port

Mode: Port FXS

Command	Purpose
<code>node(prt-fxs)[slot/port]# [no] shutdown</code>	Shuts down or activates the port. Mandatory: You must activate the port in order to place and receive calls. By default, the port is shut down. On shutdown, active calls on that port are dropped and no more calls can be placed.

Note To be able to place calls you also have to bind the FXS port to an FXS interface (see below).

Note FXS Ports or used FXS profiles can be re-configured while the port is active. All changes are applied immediately even if a call is in place. Some configuration changes however exhibit side effects as listed below:

- When unbinding or re-binding the FXS port from an FXS interface an active call is dropped
- When changing any MWI option in the FXS profile, the MWI state is lost (i.e. the device forgets that messages are waiting). This is the case if the FXS port is shut down or activated as well.

Bind FXS Ports to FXS Interfaces in the CS Context

An FXS port needs to be associated with an FXS interface in a CS context in order to receive or place calls. The same mechanism of *binding* is used on other ports (e.g. Ethernet), see section “[Interfaces, Ports, and Bindings](#)” on page 60. FXS ports are not bound to an FXS interface by default. However, your device may be shipped with a configuration that binds the FXS ports to pre-created FXS interfaces. If this is not the case, please make sure to create your own FXS interfaces in the CS context and bind the FXS port to its corresponding interface.

Procedure: To bind/unbind an FXS port to an FXS interface in the CS context

Mode: Port FXS

Command	Purpose
<code>node(prt-fxs)[slot/port]# [no] bind interface [context] interface</code>	<p>Binds the FXS port to an <i>interface</i> in a CS context. If the context argument is omitted, the port is bound to the specified <i>interface</i> in CS context <i>SWITCH</i>.</p> <p>Mandatory: You must bind the port in order to place and receive calls. By default, the port is not bound to an interface.</p>

Note When unbinding or re-binding an FXS port to an FXS interface an active call is dropped.

Note Only one FXS port can be bound to the same FXS interface. That is, you have to create a dedicated FXS interface for each FXS port. If you want to create a trunk group where calls may use any free FXS line, create a hunt-group service in the session router (refer to section “[Creating a Hunt Group Service](#)” on page 605). If you want to create a distribution group where many phones ring for an incoming call, create a distribution-group service in the session router (refer to section “[Creating a Distribution Group Service](#)” on page 616).

Configure a Subscriber Number

You should configure a subscriber number for each FXS port. The subscriber number is used as calling-party number for outgoing calls (calls from the analog phones to other protocols such as ISDN or SIP).

Unlike ISDN where each terminal knows its own subscriber number (MSN), an analog phone does not have this capability. If a phone is connected to an FXS port and makes an outgoing call (goes off hook and dials), the dialed digits form the called-party number. But there is no calling-party information available from the phone. Instead the Trinity device uses the number configured with the *subscriber-number* command on the FXS port to build the calling-party number.

Procedure: To assign a subscriber number to an FXS port

Mode: Port FXS

Command	Purpose
<code>node(prt-fxs)[slot/port]# [no reset] subscriber-number number</code>	Configures the subscriber number for the current FXS port. The <i>number</i> argument is an arbitrary-length string consisting of the characters 0-9, *, or #. Recommended: Only if a subscriber number is configured the calling-party number will be announced for outgoing calls. No subscriber number is configured for the attached phone by default.

Note When changing the subscriber number while a call is connected, the new subscriber number is only used as calling-party number for the next call. However, if the subscriber number is changed when the user is in progress of dialing a number, the new number is used when finally placing the call to the far end.

Configure Phone-Specific Parameters

Unlike ISDN, POTS is heavily country specific even though there is a good chance that a phone for one country works reasonably well in another region of the world. As depicted in Figure 1, country- and other phone-specific settings are contained in a so-called FXS profile.

There are two built-in FXS profiles:

- **DEFAULT_EU:** Template for most European phones following the ETSI standard.
- **DEFAULT_US:** Template for most Northern American phones following the Bell standard.

The two built-in profiles provide good default values that work for most phones. However, please study the configuration options and fine-tune the used profile or create a new profile to fully support the features of the connected phone.

An FXS profile configures the following phone-specific settings:

- **Electrical standard:** Selects a country or region for which the connected phone is built.
- **Caller ID:** Specifies whether or not information about the calling party is sent to the phone and which method is used to do so.

- **Connect Signal:** If instead of a phone a PBX is connected to an FXS port, the PBX may want to receive a signal if the far-end picked up the receiver. Different methods are available such as reverting the polarity or sending a DTMF signal or a tax pulse.
- **Disconnect Signal:** In addition to the connect-signal the Patton device may break the loop for a short period to signal that the far-end disconnected the call.
- **Tax-Pulse Modulation:** If the network charges for individual phone calls the Patton device is able to forward or generate charging information to the phone by sending tax pulses (bursts of 12kHz or 16khz tones).
- **Message-Waiting Indication (MWI):** If the Patton device is active on behalf of a PBX that provides voice-mail, the device is able to forward indications for waiting messages to the phone. Again, there are many standards with which a phone may want to get notified, including polarity-reversal, sending a voltage pulse, FSK (frequency-shift keying), or playing a stuttered dial-tone when the user picks up the receiver.

Note Some of the settings combined in the FXS profile are country-specific (e.g. the electrical standard) while others are specific for different types of phones (e.g. the MWI signaling method).

Follow the tasks described below to create a new or modify an existing FXS profile and use it on the FXS ports:

- Plan your FXS profiles
- Use an existing FXS profile on an FXS port
- Create, modify, reset, or delete an FXS profile

Plan your FXS profiles

The same FXS profile can be used on multiple FXS ports if the connected phones exhibit the same electrical and physical service characteristics. If all connected phones are of the same type it is sufficient to create a single FXS profile or modify one of the built-in profiles. However, if you connect different types of phones to the same device, you most probably have to create more than one profile, i.e., one for each phone brand.

If you connect the same type of phones we recommend to use and modify one of the existing built-in profiles (*DEFAULT_EU*, *DEFAULT_US*). The built-in profiles should enable basic calls for most phone brands. However, some of the advanced services are disabled by default. Please consult the default values of the two built-in profiles in [table 36](#) to verify *whether* you have to modify the chosen profile to fully support the connected phones.

Table 36. Default settings for built-in and manually-created FXS profiles

FXS-Profile Feature	Built-In FXS profile DEFAULT_EU	Built-In FXS profile DEFAULT_US	New FXS profiles (same as DEFAULT_EU)
Electrical Standard	etsi	bell	etsi
Caller ID	fsk-etsi, mid-ring presentation	fsk-bell mid-ring presentation	fsk-etsi, mid-ring-presentation
Connect Signal	none	none	none
Disconnect Signal	none	none	none
Tax-Pulse Modulation	12-kHz bursts	16-kHz bursts	12-kHz bursts

Table 36. Default settings for built-in and manually-created FXS profiles (Continued)

FXS-Profile Feature	Built-In FXS profile DEFAULT_EU	Built-In FXS profile DEFAULT_US	New FXS profiles (same as DEFAULT_EU)
Message-Waiting Indication (MWI)	none	none	none

Note If you create new profiles next to the built-in *DEFAULT_EU* and *DEFAULT_US* profiles, the new profile inherits all configuration parameters from the *DEFAULT_EU* profile (see last column in [table 36](#)).

Note If you connect more than one type of phone we recommend to create new profiles and name them according to the phone brand.

The table below provides an overview of all configuration commands available to configure the FXS profile.

Page	Command	Purpose
11	<code>node(cfg)# [no reset] profile fxs name</code>	If there was no FXS profile called <i>name</i> , the command creates a new profile. The command always enters the profile configuration mode. The reset prefix resets the entire profile back to the default setting as listed in table 36 .
11	<code>node(pf-fxs)[name]# [reset] electrical-standard standard</code>	Selects a country-specific <i>standard</i> for driving the electrical signals (e.g. ring voltage, etc.). Optional: Modify if the phone requires a different standard than the default listed in table 37 .
12	<code>node(pf-fxs)[name]# [no reset] caller-id standard standard presentation (pre-ring mid-ring)</code>	Enables or disables the transmission of the Caller-ID. Specifies which <i>standard</i> to use and whether the caller-id is sent before (<i>pre-ring</i>) or after (<i>mid-ring</i>) the first ring. Optional: Modify if the phone requires a different caller-id standard and/or presentation than the default listed in table 37 .
13	<code>node(pf-fxs)[name]# [no reset] connect-signal (polarity-reversal dtmf-c tax-pulse)</code>	Enables/Disables the transmission of a connect signal. If enabled the device either reverses the DC polarity (and establishes the original polarity when the call is disconnected), sends a DTMF C tone, or sends a tax pulse according to the <i>tax-pulse-modulation</i> command. Optional: Typically, the connect-signal is only required if you connect a PBX instead of a phone to the FXS port that uses this profile.

Page	Command	Purpose
13	<code>node(pf-fxs)[name]# [no reset] disconnect-signal loop-break milliseconds</code>	Enables/Disables the transmission of a disconnect signal. If enabled the device drops the loop current for the specified number of milliseconds when the call is disconnected Optional: Typically, the disconnect-signal is only required if you connect a PBX instead of a phone to the FXS port that uses this profile.
15	<code>node(pf-fxs)[name]# [reset] tax-pulse-modulation (bursts-12khz bursts-16khz)</code>	Configures whether to use 12kHz or 16kHz tones to send tax pulses Optional: Modify if the phone requires a different tax-pulse modulation than the default listed in table 37 .
16	<code>node(pf-fxs)[name]# [no reset] message-waiting-indication stutter-dial-tone</code> <code>node(pf-fxs)[name]# [no reset] message-waiting-indication frequency-shift-keying</code> <code>node(pf-fxs)[name]# [no reset] message-waiting-indication pulsing (polarity-revseral voltage)</code>	Configures which methods to use to send Message-Waiting Indication (MWI) signals to the phone. The following three methods can be used in parallel. This is why there are three commands that can be enabled/disabled individually: <ul style="list-style-type: none"> • stutter-dial-tone: If there are new messages when the user picks up the receiver, the dial-tone will be interrupted repeatedly for 3 seconds. • frequency-shift-keying: If there are new messages the device sends a frequency-modulated signals in on-hook state. • pulsing: If there are new messages the device either repeatedly reverses the DC polarity or emits voltage pulses in on-hook state. Optional: Enable if the connected phone is able to accept one of the supported MWI methods.

Example: Configuration of two configured FXS ports, both using a created profile for the connected PBX requiring connect and disconnect signals and supporting MWI signals. The ports are bound to an FXS interface each, which routes the calls between the two ports (subscriber numbers 100 and 101, respectively).

```
profile fxs MY_PHONE
  electrical-standard bell
  caller-id standard fsk-bell presentation mid-ring
  connect-signal dtmfc
  disconnect-signal loop-break 250
  tax-pulse-modulation bursts-16khz
  message-waiting-indication stutter-dial-tone
  message-waiting-indication frequency-shift-keying

context cs SWITCH
  no shutdown
```

```

routing-table called-e164 RT-MAIN
  route 100 dest-interface IF-FXS-00
  route 101 dest-interface IF-FXS-01

interface fxs IF-FXS-00
  route call dest-table RT-MAIN

interface fxs IF-FXS-01
  route call dest-table RT-MAIN

port fxs 0 0
  subscriber-number 100
  use profile fxs MY_PHONE
  bind interface SWITCH IF-FXS-00
  no shutdown

port fxs 0 1
  subscriber-number 101
  use profile fxs MY_PHONE
  bind interface SWITCH IF-FXS-01
  no shutdown

```

Use an existing FXS profile on an FXS port

By default, all FXS ports use the built-in FXS profile *DEFAULT_EU*, which works well for most European phones. If you have to create a new profile, please follow the procedure in the next sub-section. To use an existing FXS profile on an FXS port, please follow the procedure below.

Procedure: To use an FXS profile on an FXS port

Mode: Port FXS

Command	Purpose
<code>node(prt-fxs)[slot/port]# [reset] use profile fxs name</code>	Associates the FXS profile <i>name</i> with the current FXS port <i>slot/port</i> . That is, the phone-characteristics configured in the profile are applied to this FXS port.

Note If you delete an FXS profile all FXS ports that currently use that profile will be reconfigured to use the default profile *DEFAULT_EU*.

Create, modify, reset, or delete an FXS profile

Please follow the steps below to create a new FXS profile, to modify an existing profile (whether it is a built-in profile or a profile you have created before), reset a profile to its default settings, or to delete a profile. The default values for built-in and new profiles are listed in [table 36](#) on page 891.

Procedure: To create, reset, or delete a new FXS profile or enter an existing one

Mode: Configure

Command	Purpose
<code>node(cfg)# [no reset] profile fxs name</code>	<p>If there was no FXS profile called <i>name</i>, the command creates a new profile. The command always enters the profile configuration mode.</p> <p>no: Deletes an existing profile. Note that the two built-in profiles <i>DEFAULT_EU</i> and <i>DEFAULT_US</i> cannot be deleted.</p> <p>reset: Resets all features of the FXS profile to their default settings according to table 36 on page 891.</p>

Each of the following sub-sections describe how to configure one feature of the FXS profile (one sub-section for each row in [table 36](#) on page 891).

Note FXS profiles can be re-configured even if they are used by active FXS ports; most changes are applied immediately even if a call is in place. However, if you change any MWI option the MWI state is lost (i.e. the device forgets that messages are waiting).

Configure the Electrical Standard

The electrical standard defines how the Patton device drives the electrical signals such as ring voltage, etc. Please make sure that you select the correct standard for your location and/or for the type of phone you connect to the port using this profile. Otherwise the Patton device will not be able to comply with your country's regulations.

Procedure: To configure the standard used to drive the electrical signals

Mode: Profile FXS

Command	Purpose
<code>node(pf-fxs)[name]# [reset] electrical-standard standard</code>	<p>Selects a country-specific standard for driving the electrical signals (e.g. ring voltage, etc.).</p> <p>Optional: Modify if the phone requires a different standard than the default listed in table 37.</p>

Table 37. Country-Codes to select the FXS electrical standard

ISO3166-1-Alpha-2 Country Code	Description
<i>au</i>	Australia
<i>bell</i>	Default configuration for Northern America
<i>ch</i>	Switzerland
<i>etsi</i>	Rest of Europe (ETSI EG 201 188)

Table 37. Country-Codes to select the FXS electrical standard (Continued)

ISO3166-1-Alpha-2 Country Code	Description
<i>gb</i>	Great Britain
<i>it</i>	Italy
<i>mx</i>	Mexico
<i>nl</i>	Netherlands
<i>nz</i>	New Zealand
<i>za</i>	South Africa

If you don't find the country of the phone brand in this list, try the generic *etsi* or *bell* configurations first. If neither works, please contact Patton support.

Note The configured electrical standard also covers the voltage applied for ringing. The actual ring pattern however is determined by the ring-cadence profile used on the FXS interface (see chapter 64, “[FXS Interface Configuration](#)” on page 823).

Note If the electrical standard is changed during an active call, the change is not applied immediately but only for the next call.

Configure the Caller-ID Presentation

Patton devices support the presentation of the Caller-ID to a connected phone while it is ringing. The Caller-ID includes the date and time, the calling-party number, and the calling-party name if available. The command below configures whether or not the Caller-ID is presented to the phone connected to the FXS port that uses this profile. The Patton device supports both FSK (frequency-modulated) transmission as well as sending sequences of DTMF signals (Dual-Tone Multi-Frequency). You also have to specify whether the Caller-ID is sent before or after the first ring signal, which depends on the connected phone.

Procedure: To configure how the Caller-ID is presented to connected phones

Mode: Profile FXS

Command	Purpose
<code>node(pf-fxs)[name]# [no reset] caller-id standard <i>standard</i> presentation (pre-ring mid-ring)</code>	Enables or disables the transmission of the Caller-ID. Specifies which <i>standard</i> to use (see Table 38 on page 897) and whether the caller-id is sent before (<i>pre-ring</i>) or after (<i>mid-ring</i>) the first ring. Optional: Modify if the phone requires a different caller-id standard and/or presentation than the default listed in table 37 .

Caller-ID standards starting with *fsk* use frequency modulation whereas standards starting with *dtmf* use dual-tone sequences to send the Caller-ID. The Caller-ID includes the calling E.164 number, the calling name if available, and the current date and time. [Table 38](#) on page 897 lists all available Caller-ID standards.

Table 38. Caller-ID standards

Caller-ID Standards	Description
<i>dtmf-brazil</i>	DTMF sequence for Brazil (Anatel 220-250-713)
<i>dtmf-denmark</i>	DTMF sequence for Denmark (TDK-TS 900 216)
<i>dtmf-etsi</i>	DTMF sequence for the rest of Europe (ETS 300 659-1 Annex C)
<i>dtmf-india</i>	DTMF sequence for India (NO.SR/ASF-01/03)
<i>fsk-etsi</i>	Frequency modulation for Europe
<i>fsk-bell</i>	Frequency modulation for Northern America

Note The Patton device currently does not support Type II Caller-ID, i.e. information about a waiting call while the user is already on the phone.

Note The configuration is applied immediately to FXS ports that use the FXS profile.

Configure Connect and Disconnect Signals

Analog phones do not need connect and disconnect signals. The user that dials a number knows when a call is connected or disconnected from the fact that the far-end speaks to him/her. However, if the FXS port is connected to a PBX or another non-human party such as an answering machine or an automated call-center, the connected device requires a signal to indicate when the far-end picked up the phone (connect signal) or hangs up (disconnect signal).

Patton devices are able to forward connect/disconnect signals received from other call protocols (e.g. SIP, ISDN) to the FXS ports. Unfortunately, there is no unique way how these signals are applied to the analog lines. Use the following procedures to configure the connect and disconnect signals:

Procedure: To configure how connect-events are signaled to the attached PBX or phone

Mode: Profile FXS

Command	Purpose
<code>node(pf-fxs)[name]# [no reset] connect-signal (polarity-reversal dtmf-c tax-pulse)</code>	<p>Enables/Disables the transmission of a connect signal. If enabled the device either reverses the DC polarity (and establishes the original polarity when the call is disconnected), sends a DTMF C tone, or sends a tax pulse according to the <i>tax-pulse-modulation</i> command (see table 39 on page 898).</p> <p>Optional: Typically, the <i>connect-signal</i> is only required if you connect a PBX instead of a phone to the FXS port that uses this profile.</p>

Procedure: To configure how disconnect-events are signaled to the attached PBX or phone

Mode: Profile FXS

Command	Purpose
<code>node(pf-fxs)[name]# [no]reset disconnect-signal loop-break milliseconds</code>	<p>Enables/Disables the transmission of a disconnect signal. If enabled the device drops the loop current for the specified number of milliseconds when the call is disconnected (see table 39).</p> <p>Optional: Typically, the disconnect-signal is only required if you connect a PBX instead of a phone to the FXS port that uses this profile.</p>

Table 39. Connect and disconnect signals

Connect-Signal	Disconnect-Signal	Description
<i>polarity-reversal</i>	<i>polarity-reversal</i>	<p>Reverses the DC polarity when the call is connected; reverses the line polarity again to normal polarity when the call is disconnected. This is often also referred to as battery-reversal (swap DC feed of tip and ring).</p> <p>Note You cannot configure the <i>disconnect-signal</i> to <i>polarity-reversal</i> explicitly; configuring the <i>connect-signal</i> to <i>polarity-reversal</i> implicitly configures the <i>disconnect-signal</i> to <i>polarity-reversal</i>, too.</p>
<i>dtmf-c</i>		Plays a DTMF C signal when the call is connected.
<i>tax-pulse</i>	<i>dtmf-etsi</i>	Sends a tax pulse when the call is connected; the <i>tax-pulse</i> method must be configured with the <i>tax-pulse-modulation</i> command (see below).
	<i>loop-break</i>	Drops the loop current for a short (configurable) period

Note Although not configured explicitly the Patton device always plays a call-progress tone when the call is disconnected. Dependent on the disconnect reason this may be the busy-tone (if a call-attempt reaches a busy user), the release-tone (if the far-end user disconnects a connected call) or a confirmation-tone (if the phone successfully activated a service). These call-progress tones are played whether or not the loop-break disconnect-signal is enabled or disabled.

Note If *polarity-reversal* is used to signal connect and disconnect events and if in addition *loop-break* is configured as *disconnect-signal*, the Patton device first restores the polarity and then breaks the loop current for the configured period.

Note Changes to the connect- and disconnect-signals are applied immediately. There is however one exception: If the connect-signal is change to or from

polarity-reversal while a call is active, the disconnect-signal of that call is still performed as configured when the call was established.

Configure Tax-Pulse Modulation

Advice-of-Charge (AOC) information received from ISDN or SIP interfaces can be forwarded to analogue FXS lines by sending bursts of 12-kHz or 16-kHz tones. In addition, the Patton device is able to generate tax pulses locally. The AOC configuration consists of three parts:

- 1 The FXS profile (current chapter) only configures the physical characteristics of emitting tax pulses, i.e., whether tax-pulses are sent as 12-kHz or 16-kHz tones.
- 2 The FXS Supplementary-Services profile (see section “Advice-of-Charge (AOC) Tax-Pulses” on page 868) configures the logical aspects of tax pulses, i.e., whether or not charge information received from SIP or ISDN is emitted to the FXS line, how charge information is converted into tax pulse, and whether or not tax pulses are generated locally.
- 3 The peer protocol (SIP, ISDN) must be configured to accept charge information. Refer to chapter 63, “ISDN Interface Configuration” on page 806 or chapter 52, “SIP Interface Configuration” on page 647, respectively.

The following procedure configures the physical characteristics of emitting tax pulses:

Procedure: To configure the modulation method for tax pulses

Mode: Profile FXS

Command	Purpose
<code>node(pf-fxs)[name]# [reset] tax-pulse-modulation (bursts-12khz bursts-16khz)</code>	Configures whether to use 12-kHz or 16-kHz tones to send tax pulses. Optional: Modify if the phone requires a different tax-pulse modulation than the default listed in table 37 .

Patton devices support the methods listed in [table 40](#) to signal tax pulses to the connected phone.

Table 40. Tax-Pulse Modulation Methods

Tax-Pulse Modulation Methods	Description
<code>bursts-12khz</code>	Sends a short 12-kHz tone for each tax pulse; this is the most common method understood by European phones according to the ETSI standard.
<code>bursts-16khz</code>	Sends a short 16-kHz tone for each tax pulse; this is the most command method understood by Northern American phones according to the Bell standard.

Note Note: Changes of the tax-pulse modulation configuration are applied immediately. That is, the method may change during the call.

Configure Message-Waiting Indication (MWI) Feature for FXS

A phone connected to an FXS port may want to get informed if there are voice-mail messages waiting on a mailbox. The Trinity device does not host the voice-mails but rather forwards MWI signals from SIP or ISDN to the FXS port. Therefore the MWI feature must be configured for all involved protocols; item (1) for FXS and either item (2) for ISDN or (3) for SIP has to be configured.

- 1 The MWI configuration in the FXS profile (this chapter) configures the method with which the message-waiting signals are sent to the phone. You can choose one or multiple of the following methods:
 - a Stuttered dial-tone: If there are new voice-mail messages in the user's mailbox, for 3 seconds after picking up the receiver the dial-tone will "stutter", i.e. the dial-tone is repeatedly interrupted for a very short period.
 - b Visual MWI (VMWI): Many phones have a lamp or LED on the case that blinks when a new message is available. The Patton device uses frequency-shift keying or sends a polarity-reversal or voltage pulses to toggle the lamp.
- 2 If the Trinity device is connected to the message box over ISDN, the method for MWI reception has to be configured on the ISDN interface (see section "[Configuring the Message Waiting Indication Feature for ISDN](#)" on page 821).
- 3 If the Trinity device is connected to the message box over SIP, the method for MWI reception has to be configured on the location service (see section "[Configuring the Message Waiting Indication Feature for SIP](#)" on page 556).

The following procedure configures the method the FXS port uses to notify the phone about waiting messages; all methods can be enabled independently.

Procedure: To configure which method is used to send an indication for waiting messages to the connected phone

Mode: Profile FXS

Command	Purpose
<code>node(pf-fxs)[name]# [no reset] message-waiting-indication stutter-dial-tone</code>	Configures which methods to use to send Message-Waiting Indication (MWI) signals to the phone. The following three methods may be used in parallel. This is why there are three commands that can be enabled/disabled individually. They control what is happening if a waiting message is present: <ul style="list-style-type: none"> • stutter-dial-tone: When the user picks up the receiver, the dial-tone will be interrupted repeatedly for 3 seconds. • frequency-shift-keying: Lets a lamp or LED blink on the phone's case or displays a symbol on its LCD display by sending a frequency-modulated signal in on-hook state. • pulsing: Lets a lamp or LED blink on the phone's case or displays a symbol on its LCD display by sending polarity or voltage pulses. <ul style="list-style-type: none"> - polarity-reversal: Repeatedly reverses the DC polarity for a short time. - voltage: Repeatedly sends voltage pulses. Optional: Enable if the connected phone is able to accept one of the supported MWI methods.
<code>node(pf-fxs)[name]# [no reset] message-waiting-indication frequency-shift-keying</code>	
<code>node(pf-fxs)[name]# [no reset] message-waiting-indication pulsing (polarity-reversal voltage)</code>	

Note When changing any MWI configuration, the MWI state is lost (i.e. the Patton device forgets that messages are waiting).

Displaying the Configuration and State of an FXS port

The following command prints configuration and state information of a certain FXS port.

Procedure: To display information of an FXS port.

Mode: Operator Exec

Command	Purpose
<code>node> show port fxs slot port</code>	Displays configuration and state information of FXS port <i>slot/port</i> .

Example: Display information of FXS port 0/0.

```
>show port fxs 0 0
Port: fxs 0 0 0
=====
Logical port:          0
Physical port:        1
```

```

TDM offset:                               8
Configuration
  Enabled:                                 true
  Subscriber number:                       100
  Country specification:                   etsi
  Connect signal:                          None
  Disconnect signal:                       None
  Loop break time:                         0[ms]
  Tax-Pulse method:                        12kHz
  Tax-Pulse active time:                   90[ms]
  Tax-Pulse inactive time:                 200[ms]
  MWI stutter:                             false
  MWI FSK:                                 false
  MWI pulse mode:                          None
  MWI pulse time:                          500[ms]
  Caller-ID format:                        FSK ETSI
  Caller-ID position:                      Mid-Ring
  PCM Law:                                  A-law
Calibrated:                                true
Channel:                                    fxs 0 0 0/bearer 0
  Line state:                              VP_LINE_STANDBY
  State-Machine
    State:                                  onHookStandby
  Configuration
    Connect signal:                         None
    Disconnect signal:                      None
    MWI pulse mode:                         None
    MWI pulse time:                         500[ms]
  Status
    MWI FSK pending:                       false
    MWI pulse pending:                     false
    Update country pending:                 false
    Tax pulses to send:                     0
    Tax pulses sending:                     0

```

Table 41 explains the displayed fields.

Table 41. FXS Port Fields

FXS Port Fields	Description
<i>Logical port</i>	The port number as referred to in the configuration and as labeled next to the connector.
<i>Physical port</i>	Internal port number w.r.t. the FXS hardware chip. This port is not used when communicating to the user, e.g. in debug messages or in the configuration.
<i>TDM offset</i>	Channel number on the TDM bus
Configuration	Reflects the user configuration passed to the FXS hardware.
<i>Subscriber number</i>	Subscriber number as configured on the FXS port. The subscriber number is used as calling-party number on outgoing calls (e.g. over ISDN or SIP).
<i>Enabled</i>	true if the <i>no shutdown</i> command has been executed; false for <i>shutdown</i> .

Table 41. FXS Port Fields (Continued)

FXS Port Fields	Description
<i>Country specification</i>	According to the <i>electrical-standard</i> configuration in the used FXS profile.
<i>Connect signal</i>	According to the <i>connect-signal</i> configuration in the used FXS profile.
<i>Disconnect signal</i>	According to the <i>disconnect-signal</i> configuration in the used FXS profile.
<i>Loop break time</i>	The loop-break time for the configured <i>disconnect-signal</i> ; 0[ms] if no disconnect signal has been configured.
<i>Tax-Pulse method</i>	According to the <i>tax-pulse-modulation</i> command in the used FXS profile.
<i>Tax-Pulse active time</i>	Duration of a tax pulse (cannot be configured)
<i>Tax-Pulse inactive time</i>	Min. duration of the pause between two tax pulses (cannot be configured)
<i>MWI stutter</i>	Whether to play the stuttered dial-tone to indicate waiting messages, according to the <i>message-waiting-indication stutter-dial-tone</i> command in the used FXS profile.
<i>MWI FSK</i>	Whether to use frequency-shift-keying to send a message waiting indication, according to the <i>message-waiting-indication frequency-shift-keying</i> command in the used FXS profile.
<i>MWI pulse mode</i>	Whether to use polarity-reversal or voltage pulses to indicate waiting messages, according to the <i>message-waiting-indication pulsing</i> command in the used FXS profile.
<i>Caller-ID format</i>	Which standard to use to send the calling-party number and name to the phone, according to the <i>caller-id</i> command in the used FXS profile.
<i>Caller-ID position</i>	Whether to send the Caller-ID before or after the first ring, according to the <i>caller-id</i> command in the used FXS profile.
<i>PCM Law</i>	Whether to use A-Law or u-Law on the TDM backplane of the device. For FXS ports this law-selection has no influence as analog signals are converted either to A-Law or u-Law. For E1/T1 ports this defines the encoding of the TDM samples. This is a global device setting; it can be configured with the <i>law-select</i> command in the system mode.
Calibrated	The FXS hardware chip calibrates the analog circuitry at boot-up. This flag reports whether or not this calibration was successful.
Channel	Reports the state of a channel in the FXS hardware.
<i>Line state</i>	Line state as reported by the FXS hardware chip.
State-Machine	Reports the state of a software state-machine that services the FXS hardware chip.

Table 41. FXS Port Fields (Continued)

FXS Port Fields	Description
<i>State</i>	State of the software state machine: <ul style="list-style-type: none"> • initial: while calibrating the chip; should transition to disable immediately after boot-up • disabled: The FXS port is shut down • onHookTimer: The FXS port went on-hook shortly before and now waits for a short time before starting the next action (e.g. ringing or sending a MWI signal) • onHookStandby: regular on-hook state, waiting for an incoming call or for the user to pick-up the phone • onHookMwiFsk: on-hook state, in progress of sending a frequency-modulated message-waiting indication • onHookMwiVoltage: on-hook state, in progress of sending a voltage-pulse to indicate that messages are waiting • onHookMwiPulseOn: on-hook state, DC polarity reversed to indicate that messages are waiting (the state machine toggles between this and the next state) • onHookMwiPulseOff: on-hook state, DC polarity un-reversed to indicate that messages are waiting (the state machine toggles between this and the previous state) • onHookRinging: ring-voltage is applied to the FXS port; the connected phone should ring • offHookDisconnected: the phone is off-hook but the far-end has not connected yet or already disconnected • offHookConnected: the phone is off-hook, as is the far-end • offHookConnectedTax: the phone is off-hook as is the far-end; tax pulses are being sent right now
<i>Configuration</i>	Repeats parts of the configuration that is relevant for the state machine
<i>MWI FSK pending</i>	If the line eventually goes on-hook, we will have to send an MWI signal using frequency-shift-keying
<i>MWI pulse pending</i>	If the lines eventually goes on-hook, we will have to send an MWI signal using voltage or line-reversal pulsing
<i>Update country pending</i>	If the electrical specification changed while the phone is off-hook, this flag remembers to reload the country specification when the line eventually goes on-hook
<i>Tax pulses to send</i>	How many tax pulses still to send when the line eventually goes on-hook
<i>Tax pulses sending</i>	How many tax pulses still to send (the device is currently sending pulses, counts down the pulses remaining)

Note If the displayed information does not match the expected configuration or state, please use enable the FXS port debug monitor (see chapter 69, “[Debug and Monitoring](#)” on page 906).

Chapter 69 **Debug and Monitoring**

Chapter contents

Introduction.....	907
Debugging Strategy.....	907
Verifying IP Connectivity.....	908
Network Sniff Tool.....	908
Debugging Call Signaling.....	908
Debugging ISDN Signaling.....	909
Debugging SIP Signaling.....	909
Verify an incoming call.....	910
Verify an outgoing call.....	910
Debugging FXS Signaling.....	910
Verify an incoming call.....	911
Verify an outgoing call.....	913
Using Trinity's Internal Call Generator.....	914
Debugging the SIP Stack.....	914
Debugging Voice Data.....	915
How to Submit Trouble Reports to Patton.....	917
What is PacketSmart?.....	917
PacketSmart in Trinity.....	918
Configuring a PacketSmart Profile.....	918
What can I do with PacketSmart?.....	919

Introduction

This chapter describes how to use the internal debugging capabilities within Trinity. It provides debugging strategies to help locate the source of a problem, and describes the **show** and **debug** commands used to verify correct system operation and to troubleshoot problems.

This chapter includes the following sections:

- Debugging strategy (see [page 907](#))
- Verifying IP connectivity (see [page 908](#))
- Debugging call signaling (see [page 908](#))
- Debugging voice data (see [page 914](#))
- Debugging the SIP Stack (see [page 914](#))

Debugging Strategy

Multi-service IP networks are highly comprised of sophisticated systems and protocols that offer a great many possibilities. Unfortunately, the possible sources of trouble are almost as many, so it is important to use a very methodical approach when tracking down a problem:

- Work from the bottom to the top of the protocol stack. Always start by checking your physicals. Verify that cables and connectors are in good shape, verify the link layer, and check IP connectivity before working on application problems.
- Work from the core to the edge. Problems always show up end-to-end, the phone does not ring, or the browser cannot find the web site. To track down network problems it is however helpful to start with a minimal number of hops, make sure everything is OK and then increase the end-to-end distance hop by hop.



IMPORTANT

Enabling some or all debug monitors may degrade system performance (IP routing, call signaling). To avoid inadvertent permanent system performance degradation make sure all debug monitors are switched off once the configuration is debugged and running.

Note Event log files record warnings and other information from system components. Entries in the logs are time-stamped with the actual system time, so make sure the device time and its system time are the same. Otherwise, you will not be able to get some information from the event logs because the time stamps are unusable.

You can enter the system time manually or have it be automatically set via SNTP. Refer to chapter 10, “[Basic System Management](#)” on page 135, or chapter 33, “[SNTP Client Configuration](#)” on page 390.

Verifying IP Connectivity

This procedure describes how to use the **ping** command to test IP connectivity. It verifies that your device can communicate with such hosts as an IP phone, a registrar, and other VoIP gateways.

Use Telnet to access your device, then use the **ping** command to verify that an IP packet can be sent to, and received from, all hosts with which the device should be able to communicate (e.g. IP phone, Registrar, other VoIP gateways).

Mode: Administrator execution

Step	Command	Purpose
1	device#ping <i>ipaddress</i> [<i>number-of-packets</i>] [<i>timeout seconds</i>]	Determines whether an IP host can be contacted.

Network Sniff Tool

A new executable command sniff has been added. This command displays the network packets going through the given interface on the CLI or save them in a temporary file. The file content can be retrieved with the show capture command or it can be sent on a TFTP server using the copy capture:sniff.cap tftp://<target> command. The capture file can then be analyzed with Wireshark. In the background the command uses tcpdump 4.5.1, and we can also apply tcpdump filters using the expression option. For more detail on filters please refer to the tcpdump documentation: <http://www.tcpdump.org/manpages/pcap-filter.7.html>.

Mode: Administrator Execution

Step	Command	Purpose
1	device#sniff [<i>interface <ipInterface></i>] [<i>save-to-file [file-size <megabytes>]</i>] [<i>packets <max-packets-to-capture></i>] [<i>truncate <bytes-per-packet></i>] [<i>expression <boolean-exp></i>]	Capture as many packets as max-packets-to-capture on ipInterface and saves it to the capture file in the system. This file size is fixed to megabytes and it will overwrite itself once this limit is reached. To define how many bytes per packet should be captured, the option truncate can be used. expression option can be used to specify a tcpdump filter (see http://www.tcpdump.org/manpages/pcap-filter.7.html for more detail on filter)

Debugging Call Signaling

If calls do not connect, or disconnect during the process of connecting, there is a problem in call signaling. We suggest the following steps to debug call signaling:

- 1 Work from the call source to the call destination.
- 2 Make sure that the call enters correctly the context CS of your unit (debug the source signaling protocol, depending on where the call comes from).
- 3 Make sure that the call leaves correctly the context CS of your unit (debug the destination signaling protocol, depending on where the call goes to).

- 4 Debug call routing when the call enters the context CS, but it does not leave it. Remember that context CS must be activated (“no shutdown”) for call routing to work.

Please make yourself familiar with the context CS concept described in chapter 44, “CS Context Overview” on page 499. The following terminology will be used in this chapter:

- *Incoming call*: Call setup attempt from a call signaling protocol towards the context CS
- *Outgoing call*: Call setup attempt from the context CS towards a call signaling protocol

A basic call from an ISDN terminal connected to your unit over SIP to another SIP gateway is thus an *incoming* and *outgoing* call at the same time, from the context CS perspective: It *comes in* from ISDN and *goes out* over SIP.

Debugging ISDN Signaling

Overview: ISDN debug monitors

Step	Command	Purpose
1	debug isdn-signaling [<detail-level> [0-5] full-detail]	Prints all ISDN layer 3 signaling messages, and call control related activity of the ISDN interface. This is a good monitor to start with when debugging ISDN.
2	debug isdn-datapath [<detail-level> [0-5] full-detail]	Prints operations on the ISDN part of the voice data path. Use this monitor if you experience problems in the data path (no speech connectivity, speech only in one direction). This monitor is not needed to debug signaling.
3	debug isdn-control [<detail-level> [0-5] full-detail]	Enables/Disables call-control ISDN control trace

Debugging SIP Signaling

Mode: Administrator Execution

Step	Command	Purpose
1	debug sip-datapath [<detail-level> [0-5] full-detail]	Logs information related to the media channels
2	debug sip-registration [<detail-level> [0-5] full-detail]	Logs information about user registration activities
3	debug sip-signaling [<detail-level> [0-5] full-detail]	Logs call signaling related information
4	debug sip-transport [<detail-level> [0-5] full-detail]	Logs all SIP messages sent or received over the IP network
5	debug sip-location-service [<detail-level> [0-5] full-detail]	Logs information about the SIP Location-Service database

Verify an incoming call

Make sure that an incoming call from the SIP network enters correctly context CS. The following sequence shows a working call setup.

```
unit(cfg)#debug sip-transport
unit(cfg)#18:53:40 SIP_TR> Received INVITE sip:50@172.16.32.32 SIP/2.0
18:53:40 SIP_TR> Sent SIP/2.0 100 Trying
18:53:40 SIP_TR> Sent SIP/2.0 180 Ringing
18:53:43 SIP_TR> Sent SIP/2.0 200 OK
18:53:43 SIP_TR> Received ACK sip:50@172.16.32.32:5060 SIP/2.0
```

Explanation:

- The line **18:53:40 SIP_TR> Received INVITE sip:50@172.16.32.32 SIP/2.0** indicates that the *INVITE* message has been received. This means that the SIP network is functional
- **18:53:40 SIP_TR > Sent SIP/2.0 100 Trying** and **18:53:40 SIP_TR> Sent SIP/2.0 180 Ringing** indicate that responses are sent back to the SIP network. This means that the call routing is working correctly, and the call has found its destination on the gateway that is debugged. If there are no responses, or a negative response, continue debugging call routing and the destination protocol.

Verify an outgoing call

Make sure that an outgoing call from context CS leaves correctly to the SIP network. The following sequence shows a working call setup.

```
unit(cfg)#debug sip-transport
unit(cfg)#18:59:07 SIP_TR> Sent INVITE sip:60@172.16.32.33 SIP/2.0
18:59:07 SIP_TR> Received SIP/2.0 100 Trying
18:59:07 SIP_TR> Received SIP/2.0 180 Ringing
18:59:10 SIP_TR> Received SIP/2.0 200 OK
18:59:10 SIP_TR> Sent ACK sip:60@172.16.32.33:5060 SIP/2.0
```

Explanation:

- The line **18:59:07 SIP_TR> Sent INVITE sip:60@172.16.32.33 SIP/2.0** indicates that the *INVITE* was sent. Thus, call routing worked in context CS and the message left to the SIP network.
- **18:59:07 SIP_TR > Received SIP/2.0 100 Trying** indicate that responses are received from the SIP network. This means that IP connectivity is OK and the remote gateway can be reached. If there are no responses, or negative ones, continue debugging the remote SIP gateway.

Debugging FXS Signaling

Procedure: To enable FXS debug monitors

Mode: Operator Exec

Command	Purpose
<code>node> debug fxs [slot port] [detail level full-detail]</code>	Prints all operations of all or the selected FXS port (low-level). If the <i>slot</i> and <i>port</i> arguments are provided the debug monitor is only enabled for the specified port; otherwise it is enabled for all ports. The <i>detail level</i> specifies the level of detail to print, <i>full-detail</i> prints all information.
<code>node> debug fxs-signaling [detail level full-detail]</code>	Prints all operations on the FXS interfaces (high-level). The <i>detail level</i> specifies the level of detail to print, <i>full-detail</i> prints all information.

Verify an incoming call

Make sure that the FXS port is enabled and bound to an FXS interface in CS context. In this example, a POTS phone connected to FXS port 0/0 goes off hook and places a call (dials a three-digit number). The port is bound to FXS interface *IF-FXS-00*. The debug output below shows a normal (working call setup sequence.

```

15:57:28.000 FXS # [fxs 0 0 0/bearer 0] onMessageReceived: 6001
15:57:28.000 FXS-SIG # [EP IF-FXS-00] State IDLE, event OFF_HOOK
15:57:28.000 FXS-SIG # [EP IF-FXS-00] Change state to DIAL-TONE.
15:57:28.000 FXS-SIG # [EP IF-FXS-00] Change datapath direction to Receive Only.
15:57:28.051 FXS # INFO [fxs-chip 0 0 0]: Vp886GetTimestamp - read 29971
15:57:28.051 FXS # HOOK [fxs 0 0 0]: Hook Buffer 0D 05 CE ED, ts 29971
15:57:28.051 FXS # HOOK [fxs 0 0 0]: Vp886ProcessHookEvent - OFFHOOK, ts 29957
15:57:28.051 FXS # INFO [fxs-chip 0 0 0]: VpUpdateTimersInt: old 219304983, new
219378963, elapsed 73980
15:57:28.051 FXS # INFO [fxs-chip 0 0 0]: VpRemoveTimer - Timer 5 was not removed
from the queue
15:57:28.051 FXS # INFO [fxs-chip 0 0 0]: Vp886GetEvent - returned event (1)
15:57:28.051 FXS:0/0 # [fxs 0 0 0/bearer 0] State: onHookStandby | Event: ChipOff-
HookSig
15:57:28.051 FXS:0/0 # [fxs 0 0 0/bearer 0] Notify off hook
15:57:28.051 FXS:0/0 # [fxs 0 0 0/bearer 0] Set line state VP_LINE_TALK
15:57:28.051 FXS # LINE_STATE [fxs 0 0 0]: Vp886SetLineStateFxs - state 4
15:57:28.052 FXS # LINE_STATE [fxs 0 0 0]: Vp886SetLineStateFxsInt - state 4
15:57:28.052 FXS # INFO [fxs-chip 0 0 0]: Vp886GetTimestamp - read 29971
15:57:28.052 FXS # INFO [fxs-chip 0 0 0]: VpUpdateTimersInt: old 219378963, new
219378963, elapsed 0
15:57:28.052 FXS # INFO [fxs-chip 0 0 0]: VpRemoveTimer - Timer 12 was not removed
from the queue
15:57:28.052 FXS # LINE_STATE [fxs 0 0 0]: Setting system state 0x23
15:57:28.052 FXS:0/0 # [fxs 0 0 0/bearer 0] State: onHookStandby -> offHookDiscon-
nected
15:57:28.000 FXS-SIG # [EP IF-FXS-00] Play tone: dial-tone (profile: ""
15:57:28.000 FXS-SIG # [EP IF-FXS-00] Notify peer property changed (active)
15:57:28.000 FXS-SIG # [EP IF-FXS-00] Notify active peer property changed
15:57:28.000 FXS-SIG # [EP IF-FXS-00] State DIAL-TONE, event PEER_TRYING
15:57:29.000 FXS-SIG # [EP IF-FXS-00] State DIAL-TONE, event DIGIT
15:57:29.000 FXS-SIG # [EP IF-FXS-00] Change state to DIALING.
15:57:29.000 FXS-SIG # [EP IF-FXS-00] Stop tone.
15:57:29.000 FXS-SIG # [EP IF-FXS-00] Notify peer property changed (active)

```

```

15:57:29.000 FXS-SIG # [EP IF-FXS-00] Notify active peer property changed
15:57:29.000 FXS-SIG # [EP IF-FXS-00] State DIALING, event PEER_TRYING
15:57:29.000 FXS-SIG # [EP IF-FXS-00] State DIALING, event DIGIT_UPDATE
15:57:29.000 FXS-SIG # [EP IF-FXS-00] State DIALING, event DIGIT
15:57:29.000 FXS-SIG # [EP IF-FXS-00] Notify peer property changed (active)
15:57:29.000 FXS-SIG # [EP IF-FXS-00] Notify active peer property changed
15:57:29.000 FXS-SIG # [EP IF-FXS-00] State DIALING, event PEER_TRYING
15:57:30.000 FXS-SIG # [EP IF-FXS-00] State DIALING, event DIGIT_UPDATE
15:57:30.000 FXS-SIG # [EP IF-FXS-00] State DIALING, event DIGIT
15:57:30.000 FXS-SIG # [EP IF-FXS-00] Notify peer property changed (active)
15:57:30.000 FXS-SIG # [EP IF-FXS-00] Notify active peer property changed
15:57:30.000 FXS-SIG # [EP IF-FXS-00] State DIALING, event PEER_TRYING
15:57:30.000 FXS-SIG # [EP IF-FXS-00] Notify peer property changed (active)
15:57:30.000 FXS-SIG # [EP IF-FXS-00] Notify active peer property changed
15:57:30.000 FXS-SIG # [EP IF-FXS-00] State DIALING, event PEER_ALERTING
15:57:30.000 FXS-SIG # [EP IF-FXS-00] Play tone: ringback-tone (profile: "")
15:57:30.000 FXS-SIG # [EP IF-FXS-00] State DIALING, event DIGIT
15:57:30.000 FXS-SIG # [EP IF-FXS-00] State DIALING, event DIGIT_UPDATE
15:57:41.000 FXS-SIG # [EP IF-FXS-00] Notify peer property changed (active)
15:57:41.000 FXS-SIG # [EP IF-FXS-00] Notify active peer property changed
15:57:41.000 FXS-SIG # [EP IF-FXS-00] State DIALING, event PEER_CONNECTED
15:57:41.000 FXS-SIG # [EP IF-FXS-00] Change state to CONNECTED.
15:57:41.000 FXS-SIG # [EP IF-FXS-00] Stop tone.
15:57:41.000 FXS-SIG # [EP IF-FXS-00] Change datapath direction to Send/Receive.
15:57:41.541 FXS:0/0 # [fxs 0 0 0/bearer 0] Signal connect
15:57:41.541 FXS:0/0 # [fxs 0 0 0/bearer 0] State: offHookDisconnected | Event:
AppConnectSig
15:57:41.541 FXS:0/0 # [fxs 0 0 0/bearer 0] State: offHookDisconnected ->
offHookConnected

```

Explanation:

- **HOOK [fxs 0 0 0]: Vp886ProcessHookEvent – OFFHOOK:** The hardware detected that the phone went off-hook.
- **[EP IF-FXS-00] State IDLE, event OFF_HOOK:** The FXS interface recognized the phone going off-hook.

Note This line is printed before the hardware's line. This may happen because the two lines are printed by different tasks, which may disrupt the order.

- **[EP IF-FXS-00] Play tone: dial-tone:** The dial-tone is played.
- **[EP IF-FXS-00] State DIAL-TONE, event DIGIT:** The user pressed the first digit.
- **[EP IF-FXS-00] Stop tone:** This stops the dial-tone.
- **[EP IF-FXS-00] Play tone: ringback-tone:** Dialing is complete, the call is placed, the far-end is ringing, we play the ring-back tone to the connect phone.
- **[EP IF-FXS-00] Change state to CONNECTED:** The far-end picked up the receiver; the call is connected.

Verify an outgoing call

Make sure that the FXS port is enabled and bound to an FXS interface in CS context. In this example, a POTS phone connected to FXS port 0/0 is called (rings); the user picks-up the receiver and connects the call. The debug output below shows a normal (working call setup sequence).

```

16:13:43.000 FXS-SIG # [EP IF-FXS-00] State IDLE, event OFFER
16:13:43.000 FXS-SIG # [EP IF-FXS-00] Change state to RINGING.
16:13:43.000 FXS-SIG # [EP IF-FXS-00] Start Ring.
16:13:43.000 FXS-SIG # [IF-FXS-00] Using profile DEFAULT as no match is found for
property type.
16:13:43.000 FXS-SIG # [EP IF-FXS-00] Notify peer property changed (active)
16:13:43.000 FXS-SIG # [EP IF-FXS-00] Notify active peer property changed
16:13:43.000 FXS-SIG # [EP IF-FXS-00] State RINGING, event PEER_CONNECTED
16:13:43.531 FXS:0/0 # [fxs 0 0 0/bearer 0] FXS channel add ring cadence element
'pause'
for 4000ms at step 2
16:13:43.531 FXS:0/0 # [fxs 0 0 0/bearer 0] FXS channel caller ID mid-ring added at
step
4
16:13:43.532 FXS:0/0 # [fxs 0 0 0/bearer 0] FXS channel add ring cadence element
'play'
for 1000ms at step 6
16:13:43.532 FXS:0/0 # [fxs 0 0 0/bearer 0] FXS channel add ring cadence element
'pause'
for 4000ms at step 8
>pProfileBuffer[7] 22
16:13:43.532 FXS:0/0 # [fxs 0 0 0/bearer 0] FXS channel ring cadence profile with 10
steps, looping at step 6
16:13:43.532 FXS:0/0 # [fxs 0 0 0/bearer 0] FXS channel printing ring cadence pro-
file
(30):
16:13:43.532 FXS:0/0 # [fxs 0 0 0/bearer 0] Set line state VP_LINE_RINGING
16:13:43.533 FXS:0/0 # [fxs 0 0 0/bearer 0] State: onHookStandby -> onHookRinging
16:13:43.533 FXS:0/0 # [fxs 0 0 0/bearer 0] FXS channel ring start
16:13:44.532 FXS:0/0 # [fxs 0 0 0/bearer 0] FXS channel ring pause
16:13:45.671 FXS:0/0 # [fxs 0 0 0/bearer 0] State: onHookRinging | Event:
ChipCidMoreDataSig
16:13:45.671 FXS:0/0 # [fxs 0 0 0/bearer 0] Event 'ChipCidMoreDataSig' handled in
state
'onHookRinging'
16:13:45.852 FXS:0/0 # [fxs 0 0 0/bearer 0] State: onHookRinging | Event:
ChipCidMoreDataSig
16:13:45.852 FXS:0/0 # [fxs 0 0 0/bearer 0] Event 'ChipCidMoreDataSig' ignored in
state
'onHookRinging'
16:13:48.532 FXS:0/0 # [fxs 0 0 0/bearer 0] FXS channel ring start
16:13:49.532 FXS:0/0 # [fxs 0 0 0/bearer 0] FXS channel ring pause
16:13:52.000 FXS # [fxs 0 0 0/bearer 0] onMessageReceived: 6001
16:13:52.000 FXS-SIG # [EP IF-FXS-00] State RINGING, event OFF_HOOK
16:13:52.000 FXS-SIG # [EP IF-FXS-00] Change state to CONNECTED.
16:13:52.000 FXS-SIG # [EP IF-FXS-00] Stop ring.
16:13:52.000 FXS-SIG # [EP IF-FXS-00] Change datapath direction to Send/Receive.
16:13:52.442 FXS:0/0 # [fxs 0 0 0/bearer 0] State: onHookRinging -> offHookDiscon-
nected
16:13:52.442 FXS:0/0 # [fxs 0 0 0/bearer 0] Signal ring-stop

```

```

16:13:52.442 FXS:0/0 # [fxs 0 0 0/bearer 0] State: offHookDisconnected | Event:
AppRingStopSig
16:13:52.442 FXS:0/0 # [fxs 0 0 0/bearer 0] Event 'AppRingStopSig' ignored in state
'offHookDisconnected'
16:13:52.442 FXS:0/0 # [fxs 0 0 0/bearer 0] Signal connect
16:13:52.443 FXS:0/0 # [fxs 0 0 0/bearer 0] State: offHookDisconnected | Event:
AppConnectSig
16:13:52.443 FXS:0/0 # [fxs 0 0 0/bearer 0] State: offHookDisconnected ->
offHookConnected

```

Explanation:

- **[EP IF-FXS-00] Start Ring.** The FXS interfaces received a call and instructs the FXS port to start ringing.
- **[fxs 0 0 0/bearer 0] FXS channel add ring cadence element 'pause':** FXS port starts with the configured ring-cadence; starts with a pause.
- **[fxs 0 0 0/bearer 0] FXS channel add ring cadence element 'play':** Plays the next ring-signal in the cadence.
- **[EP IF-FXS-00] State RINGING, event OFF_HOOK:** The user picked up the receiver.
- **[fxs 0 0 0/bearer 0] Signal ring-stop:** Stop ringing immediately.
- **[EP IF-FXS-00] Change datapath direction to Send/Receive:** Activate the voice stream.

Using Trinity's Internal Call Generator

The Trinity has a powerful internal call generator that creates calls from the center of context CS. It is very useful to debug call signaling or call routing problems to verify the correct working of one call signaling protocol at a time. Calls can be placed towards any interface on context CS, or any routing table within context CS.

Step	Command	Purpose
	To make a call: device(cfg)#call <calling-e164-number> dial <remote-e164-number> dest-<interfacelser- vice> <interface or service name>	Generates a test call internally within the device with the ability to send the call out an interface or call-service. Only signaling can be generated, no RTP.
	To drop the call: device(cfg)#call <called-e164-number> drop	

Debugging the SIP Stack

The following commands will enable debugging of the SIP-Stack within Trinity VoIP Gateways

Overview: Operator execution

Step	Command	Purpose
1	device>debug sip-stack-core	Enables debug log traces to the current terminal.
2	device>debug sip-stack-core-service	Enables debug log traces to the current terminal.
3	device>debug sip-stack-framework	Enables debug log traces to the current terminal.

Step	Command	Purpose
4	device>debug sip-stack-parser	Enables debug log traces to the current terminal.
5	device>debug sip-stack-sdp	Enables debug log traces to the current terminal.
6	device>debug sip-stack-transaction	Enables debug log traces to the current terminal.
7	device>debug sip-stack-transport	Enables debug log traces to the current terminal.
8	device>debug sip-stack-useragent	Enables debug log traces to the current terminal.

Debugging Voice Data

There are several debug monitors that can help identify problems in VoIP connections. The most common VoIP problems are: voice quality problems (dropouts), fax transmission errors, no establishment of voice connection, and wrong tone or playback. Dependent on the Patton devices, the output of the different Media Gateway monitors can differ.

An overview of all available Media Gateway debug monitors is given below. Some more specific examples for debugging cases follow after that.

Overview: Media Gateway debug monitors

[no] debug mg-dejitter	Displays changes to the settings of the dejitter buffer, exceptions (underrun, overrun, packet drops) and size changes. Usage: To investigate problems related to voice quality, voice packet payload sizes, delays, jitter
[no] debug mg-control [detail full-detail]	Displays control activities on the Data Path (path of voice/fax data packets within your unit): State changes, tone start/stop, DTMF playback/detection, fax/modem detection. Usage: To investigate problems with voice connections, DTMF, tone playback, fax and modem transmissions.
[no] debug mg-rtp [detail full-detail]	Displays RTP related call parameters at call setup: local/remote IP address and port, SSRC. During operation, displays periodically updated statistics containing the number of sent and received packets, the number of lost packets. Usage: To verify that RTP packets are sent/received, and to debug network quality issues (lost packets).
[no] debug mg-switch [detail full-detail]	Displays control activities on the TDM part of the Data Path. Usage: To investigate problems with hair pinning or timeslot switching.
[no] debug mg-dsp [detail full-detail]	Displays control activities on the DSP including all call parameters (e.g. the used codec). Usage: To document the DSP parameters used for each call setup.

[no] debug mg-fax-data [detail full-detail]	<p>Traces detected Modem and Fax tones.</p> <p>Traces the flow of T.30 communication states between the fax machines.</p> <p>Decodes the IFP (Internet Fax Protocol) elements within T.38 packets.</p> <p>Displays T.38 related call parameters and operational errors like lost packets.</p> <p>Tracks the operation of the dejitter buffer used for T.38 Fax transmissions.</p> <p>Usage: To investigate problems with T.38 Fax and Modem transmissions in general.</p>
---	---

Depending on the type of problem, some debug monitors are more useful than others. Try to avoid enabling all monitors at the same time, as this generates a lot of output and can degrade system performance. The following examples show some typical debug cases and what monitors should be switched on in these cases.

Example: Debugging voice connections

Symptoms: Voice quality is bad (dropouts), the voice connection is only established in one direction or not at all, there is only noise instead of voice, no tones are played (e.g. dialtone).

Prerequisite: The call is established from a signaling point of view.

Use the following debug monitors:

Step	Command	Purpose
1	unit#debug mg-rtp	Enable the RTP/RTCP debug monitor
2	unit#debug mg-dejitter	Enable the dejitter buffer monitor.

Example: Debugging Fax connections

Symptoms: Fax transmission starts but is interrupted and the Fax machines terminate with an error message, Fax transmission does not start at all, the unit's firmware does not detect Fax.

Prerequisite: Fax transmission is configured for T.38 relay in the voip profile. The call is established from a signaling point of view (see section "[Debugging the SIP Stack](#)" on page 914).

Attention: Special signaling procedures are used for the transition between voice and fax data transmission. It may be that the initial call setup is correct, but that the signaling to T.38 over SIP is faulty. The session-control and SIP signaling monitors (see Debug Session Control Data, and Debug SIP Data) might also be helpful to identify these cases.

Use the following debug monitors if you assume that signaling is OK:

Step	Command	Purpose
1	unit#debug mg-dsp	Enable the dsp event monitor.
2	unit#debug mg-fax-data	Enable the Fax/Modem event monitor
3	unit#debug mg-control	Enable the control monitor

How to Submit Trouble Reports to Patton

Due the wealth of functionality and complexity of the products there remains a certain number of problems, either pertaining to the Patton product or the interoperability with other vendor's products.

If you have a problem for which you need supplier help please prepare and send the following information:

- **Problem description**—Add a description of the problem, if possible together with applicable augmented information with a diagram of the network setup (with Microsoft tools).
- **Running configuration and software and hardware version information**—With the Command Line Interface commands **show running-config** and **show version** you can display the currently active configuration of the system (in a Telnet and/or console session). Adding to the submitted trouble report will help us analyze the configuration and preclude possible configuration problems. In the unlikely case of a suspected hardware problem, also submit the serial number of your unit(s) and/or interface cards.
- **Event logs**—Add the system event logs, which you can display with the Command Line Interface commands **show log** and **show log supervisor**. To ensure that the logs are useful, it is necessary to set upon start up the clock to actual date and time (by hand or by enabling SNTP client)

If possible, add the following information in addition to the above:

- **Logs of protocol monitors**—Protocol traces contain a wealth of additional information which may be very helpful in finding or at least pinpointing the problem. Various protocol monitors with different levels of detail are an integral part of the firmware and can be started (in a Telnet and/or console session) individually (**debug** command).

Note In order to correlate the output of different protocol monitors (e.g. ISDN signaling and gateway SIP signaling), run the monitors concurrently. You can do this either in the same Telnet session, or using different Telnet sessions.

- **Network traffic traces**—In certain cases it may be helpful to have a trace of the traffic on the IP network in order to inspect packet contents. Please use one of the following tools (supporting trace file formats which our tools can read):
 - Network Associates Sniffer—Details are available at www.sniffer.com
 - TTC Firebird—Details are available at www.ttc.com
 - Wireshark - Details are available at www.wireshark.com (freeware)

When possible, submit the package of trouble report files by mail to the following address: support@patton.com (use fax only in exceptional cases).

What is PacketSmart?

PacketSmart is a network monitoring application which runs on probes (e.g. PI-150) installed physically in the network itself.

While it is active, PacketSmart gathers VoIP traffic information and sends statistics to a central BroadSoft server. Then, using the PacketSmart GUI on a PC desktop, we can retrieve this information in the form of graphical plots and meaningful data in order to detect any problem in the network as soon as possible. By detecting errors soon enough, service providers can then fix them before they become an issue, assuring customers a valuable quality of service.

PacketSmart in Trinity

PacketSmart is integrated in Trinity for the products SN4970, SN4980, SN4990, SN5480, SN5490, SN5300, SN4130, SN4170, SN5530, and SN5570. In order to use it, a Patton PacketSmart License needs to be purchased and activated, and the device itself needs to be registered in the BroadSoft server.

Once it is configured and running, PacketSmart will monitor all VoIP traffic that goes through an Ethernet interface. It will measure the amount of (total) traffic, gather all SIP messages for every call, calculate VoIP metric values and statistics, and upload this data to the BroadSoft server. Because more calls means more processing and PacketSmart is *heavy* on the use of resources, a monitoring SmartNode will see the maximum of calls it can handle limited by a factor that depends on the product.

Configuring a PacketSmart Profile

Configuring PacketSmart in Trinity is very simple. After activating the license and rebooting the device, a *profile packetsmart DEFAULT* will appear in the running configuration. Then it is only a few steps left to run PacketSmart:

- 1 In configure mode, access **profile packetsmart** mode.
- 2 **Bind** the PacketSmart profile to a valid IP interface to communicate to the server.
- 3 (*Optional*) Configure the URI of the BroadSoft server and the IP protocol (IPv4 or IPv6) to communicate with it.
- 4 (*Optional*) Configure SIP and RTP ports to use when generating calls for network assessment tests.
- 5 In the Ethernet port mode (or bridge-group or switch-group mode) **use** the PacketSmart profile to start monitoring.

And that's it.

Note More than one PacketSmart profile can be created, but only one can be in use at any time.

You can find the step-by-step command guide below.

Step	Command	Purpose
1	<code>node(cfg)]# profile packetsmart <pf-name></code>	Access PacketSmart profile mode
2	<code>node(pf-psmart)[<pf-name>]]# [no] bind interface <ip-if></code>	Choose an IP interface to upload data to PacketSmart server
3	<code>node(pf-psmart)[<pf-name>]]# server [<ip-or-host> [[port <port-number>] [ip-mode { ipv4 ipv6 }]]]</code>	Configure the URI of the PacketSmart server and the IP protocol (IPv4 or IPv6) to communicate with it
4	<code>node(pf-psmart)[<pf-name>]]# assessment sip port [<port>]</code>	Configure SIP port to use when generating calls for network assessment tests
5	<code>node(pf-psmart)[<pf-name>]]# assessment rtp port-range [<port-min> <port-max>]</code>	Configure RTP ports to use when generating calls for network assessment tests
6	<code>node(prt-eth)[<slot/port>]]# [no] use profile packetsmart <pf-name></code>	In Port mode (DSL/ETH) or Switch/Bridge mode, use profile packetsmart to start monitoring

Depiction of how PacketSmart commands look like in a Trinity config:

Figure 139. What PacketSmart commands look like in a Trinity config

Output of the command "show profile packetsmart":

Figure 140. Output of the command "show profile packetsmart":

Version: PacketSmart software version

Entity ID: Key ID used to register the device in the BroadSoft server

Server: BroadSoft server URI

IP mode: IP protocol used to upload data to the BroadSoft server

Bind: IP interface used to upload data to the BroadSoft server

Monitor: Ethernet interface being monitored

Assessment SIP port: SIP ports to use when generating calls for network assessment tests

Assessment RTP port range: SIP ports to use when generating calls for network assessment tests

State: One of the following

- **DOWN** - not running
- **NOT CONNECTED** - can't reach the server, or device not provisioned
- **NOT ACTIVATED** - connected and provisioned, but not activated
- **CONNECTED** - up and monitoring

What can I do with PacketSmart?

Besides monitoring 24/7, a great feature is the **VoIP network assessment**, where a PacketSmart device (source) will **generate** and monitor a number of calls to another PacketSmart device (target), and assess the performance of the network. The number of calls and their duration are configurable, and assessment tests can be easily scheduled through the PacketSmart GUI, which can be downloaded from the Broadsoft website.

A series of PacketSmart video tutorials can be found in the following link:

<https://www.youtube.com/user/packetsmart>

Further documentation for PacketSmart should be gathered from Broadsoft directly.

Chapter 70 **Contacting Patton for Assistance**

Chapter contents

- Introduction.....921
- Contact information.....921
 - Contacting Patton Technical Services for Free Support921
- Warranty Service and Returned Merchandise Authorizations (RMAs).....921
 - Warranty coverage921
 - Out-of-warranty service922
 - Returns for credit922
 - Return for credit policy922
 - RMA numbers922
 - Shipping instructions922

Introduction

This chapter contains the following information:

- “Contact information”—describes how to contact Patton technical support for assistance.
- “Warranty Service and Returned Merchandise Authorizations (RMAs)” —contains information about the warranty and obtaining a return merchandise authorization (RMA).

Contact information

Patton Electronics offers a wide array of free technical services. If you have questions about any of our other products we recommend you begin your search for answers by using our technical knowledge base. Here, we have gathered together many of the more commonly asked questions and compiled them into a searchable database to help you quickly solve your problems.

Contacting Patton Technical Services for Free Support

REGION	North America	Western Europe	Central & Eastern Europe	Middle East North Africa
Location	Maryland, USA	Bern, Switzerland	Budapest, Hungary	Beirut, Lebanon
Time Zone	EST/EDT UTC/GMT - 4/5 hours	CET/CEDT UTC/GMT + 1/2 hours	CET/CEDT UTC/GMT + 1/2 hours	EET/EEDT UTC/GMT + 2/3 hours
Business Hours	Monday-Friday 8:00am to 5:00pm	Monday-Friday 09:00 to 12:00 13:30 to 17:30	Monday-Friday 8:30 to 17:00	Monday-Friday 8:00am to 5pm
Email	support@patton.com	support@patton.com	support@patton.com	support@patton.com
Phone	+ 1 301 975 1007	+41 31 985 25 55	+36 439 3835	+96 1 359 1277
Fax	+1 301 869 9293	+41 31 985 2526		

Warranty Service and Returned Merchandise Authorizations (RMAs)

Patton Electronics is an ISO-9001 certified manufacturer and our products are carefully tested before shipment. All of our products are backed by a comprehensive warranty program.

Note If you purchased your equipment from a Patton Electronics reseller, ask your reseller how you should proceed with warranty service. It is often more convenient for you to work with your local reseller to obtain a replacement. Patton services our products no matter how you acquired them.

Warranty coverage

Our products are under warranty to be free from defects, and we will, at our option, repair or replace the product should it fail within one year from the first date of shipment. Our warranty is limited to defects in workmanship or materials, and does not cover customer damage, lightning or power surge damage, abuse, or unauthorized modification.

Out-of-warranty service

Patton services what we sell, no matter how you acquired it, including malfunctioning products that are no longer under warranty. Our products have a flat fee for repairs. Units damaged by lightning or other catastrophes may require replacement.

Returns for credit

Customer satisfaction is important to us, therefore any product may be returned with authorization within 30 days from the shipment date for a full credit of the purchase price. If you have ordered the wrong equipment or you are dissatisfied in any way, please contact us to request an RMA number to accept your return. Patton is not responsible for equipment returned without a Return Authorization.

Return for credit policy

- Less than 30 days: No Charge. Your credit will be issued upon receipt and inspection of the equipment.
- 30 to 60 days: We will add a 20% restocking charge (crediting your account with 80% of the purchase price).
- Over 60 days: Products will be accepted for repairs only.

RMA numbers

RMA numbers are required for all product returns. You can obtain an RMA by doing one of the following:

- Completing a request on the RMA Request page in the *Support* section at **www.patton.com**
- By calling **+1 (301) 975-1007** and speaking to a Technical Support Engineer
- By sending an e-mail to **returns@patton.com**

All returned units must have the RMA number clearly visible on the outside of the shipping container. Please use the original packing material that the device came in or pack the unit securely to avoid damage during shipping.

Shipping instructions

The RMA number should be clearly visible on the address label. Our shipping address is as follows:

Patton Electronics Company

RMA#: xxxx

7622 Rickenbacker Dr.

Gaithersburg, MD 20879-4773 USA

Patton will ship the equipment back to you in the same manner you ship it to us. Patton will pay the return shipping costs.

Appendix A **Trinity Architecture Terms and Definitions**

Chapter contents

Introduction.....924

Introduction

This chapter contains the terms and their definitions that are used throughout this *Trinity Release 3.0 Command Line Reference Guide*. This guide contains many terms that are related to specific networking technologies areas such as LAN protocols, WAN technologies, routing, Ethernet, and Frame Relay. Moreover various terms are related to telecommunication areas.

Term or Definition	Meaning
Administrator	The person who has privileged access to the CLI.
Application Download	A application image is downloaded from a remote TFTP server to the persistent memory (flash:) of a Patton device.
Application Image	The binary operation code stored in the persistent memory (flash:) of a Patton device.
Batchfile	Script file containing instructions to download one or more software component from a TFTP server to the persistent memory (flash: or nvram:) of a Patton device.
Bootloader	The bootloader is a “mini” application performing basic system checks and starting the application image. The bootloader also provides minimal network services allowing the Patton device to be accessed and upgraded over the network even if the application image should not start. The bootloader is installed in the factory and is in general never upgraded.
Bootloader Image	The binary code of the Bootloader stored in the persistent memory (flash:) of a Patton device.
Bootstrap	The starting-up of a Patton device, which involves checking the Reset button, loading and starting the application image, and starting other software modules, or—if no valid application image is available—the boot-loader.
Build	The released software is organized as builds. Each build has its unique identification. A build is part of a release and has software bug fixes. See also <i>release</i> .
Call Routing	Calls through Patton device can be routed based on a set of routing criteria. See also <i>Session Router</i> .
Call Signaling	The call signaling specifies how to set up a call to the destination Patton device or 3rd party equipment.
Circuit	A communication path between two or more devices.
Circuit Port	Physical port connected to a switching system or used for circuit switching.
Circuit Switching	The switching system in which a dedicated physical circuit path must exist between the sender and the receiver for the duration of the call. Used in the conventional telephone network.
Codec	Abbreviation for the word construct Coder and Decoder. Voice channels occupy 64 kbps using PCM (pulse code modulation) coding. Over the years, compression techniques were developed allowing a reduction in the required bandwidth while preserving voice quality. Such compression techniques are implemented within a Codec.

Term or Definition	Meaning
Comfort Noise	Comfort noise is generated at the remote end of the silent direction to avoid the impression that the connection is dead. See also <i>Silence Compression</i> .
Command Line Interface	An interface that allows the user to interact with the Trinity operating system by entering commands and optional arguments. Other operating systems like UNIX or DOS also provide CLIs.
Configuration Download	A configuration file is downloaded from a remote TFTP server via TFTP to the persistent memory (nvram:) or volatile memory (system:) of a Patton device.
Configuration File	The configuration file contains the CLI commands, which are used to configure the Patton device.
Configuration Server	A central server used as a store for configuration files, which are downloaded to or uploaded from a Patton device using TFTP.
Configuration Upload	A configuration file is uploaded from the persistent memory (nvram:) or volatile memory (system:) of a Patton device via TFTP to a TFTP server.
Context	Represents one specific networking technology or protocol, e.g. IP or circuit switching.
Data Port	Physical port connected to a network element or used for data transfer.
Dejitter Buffer	The element used to compensate variable network delays. Storing packets in a dejitter buffer before they are transferred to the local ISDN equipment, e.g. telephone, a variable delay is converted into a fixed delay, giving voice a better quality. See also <i>Jitter</i> .
Digit Collection	Some devices (PBX, ISDN network, and remote gateways) may require bloc sending of the dialed number. Digit collection collects the overlap dialed digits and forwards them in a single call setup message
Driver Software Download	A driver software image is downloaded from a remote TFTP server to the persistent memory (flash:) of a Patton device.
Driver Software Image	The software used for peripheral chips on the main board and optional PMC interface cards is stored in the persistent memory (flash:) of a Patton device.
DTMF Relay	DTMF relay solves the problem of DTMF distortion by transporting DTMF tones over low-bit-rate codecs out-of-band or separate from the encoded voice stream
Echo Cancellor	Some voice devices unfortunately have got an echo on their wire. Echo cancellation provides near-end echo compensation for this device.
Factory Configuration	The factory configuration (factory-config) represents the system default settings and is stored in the persistent memory (nvram:) of a Patton device.
Fast Connect	Since a normal call setup is often too slow, fast connect is a new method of call setup that bypasses some usual steps in order to make it faster.
Flash Memory	Persistent memory section of a Patton device containing the Application Image, Bootloader Image and the driver software Image.
flash:	A region in the persistent memory of a Patton device. See also <i>flash memory</i> .

Term or Definition	Meaning
Gateway	A gateway refers to a special purpose component that connects two contexts of different types, For example, the CS and the IP context. It handles connections between different technologies or protocols.
High-Pass Filter	A high-pass filter is normally used to cancel noises at the voice coder input. See also <i>post filter</i>
Host	Computer system on a network. Similar to node, except that host usually implies a PC or workstation, whereas node generally applies to any networked system, including access servers and routers. See also <i>node</i> .
Hostname	Name given to a computer system, e.g. a PC or workstation.
Hunt Group	In Patton device terminology, a hunt groups allows you to apply the interface configuration to multiple physical ports. Within the hunt groups free channels for outgoing calls are hunted on all available ports. In general a hunt group represents a group of trunk lines as used for direct dialing in (DDI).
Interface	An interface is a logical construct that provides higher-layer protocol and service information. An Interface is configured as a part of a context, and is independent of a physical port or circuit.
Interface Card	An optional plug-in card offering one or more ports of a specific physical standard for connecting the Patton device to the outside world.
ISDN	Integrated Services Digital Network
ISDN Services	ISDN Services comprise voice, data, video and supplementary services. Supplementary services are services available in the ISDN network, such as calling line identification presentation (CLIP) or call waiting (CW). See also <i>Q.SIG</i>
Jitter	Jitter is the variation on packets arriving on a Patton device. See also <i>dejitter buffer</i> .
Mode	The CLI is comprised of modes. There are two basic mode groups, the execution mode group and the configuration mode group.
Network Management System	System responsible for managing at least part of a network. An NMS is generally a reasonably powerful and well-equipped computer, such as an engineering workstation. NMSs communicate with agents to help keep track of network statistics and resources.
Node	Endpoint of a network connection or a junction common to two or more lines in a network. A Node can be a router, e.g. a Patton device. Nodes, which vary in routing and other functional capabilities, can be interconnected. Node sometimes is used generically to refer to any entity that can access a network, and frequently is used interchangeably with device.
Nodename	Name given to a Patton device or network element.
nvrnm:	Persistent memory section of a Patton device containing the startup configuration, the factory configuration and used defined configurations.
Operator	The person who has limited access to the CLI.

Term or Definition	Meaning
PCI Local Bus	The PCI Local Bus is a high performance, 32-bit or 64-bit bus with multiplexed address and data lines. The bus is intended for use as an interconnect mechanism between highly integrated peripheral controller components, peripheral add-in boards, and processor/memory systems.
PCM Highway	A 30 channel interface connecting the switching engine with optional interface cards containing circuit ports.
Port	A port represents a physical connector on the Patton device.
Port Address	A port address can be assigned to a CS interface to realize a virtual voice tunnel between two nodes.
Post Filter	The voice decoder output is normally filtered using a perceptual post-filter to improve voice quality. See also <i>High-Pass Filter</i> .
POTS	Plain Old Telephone Service
Profile	A profile provides configuration shortcutting. A profile contains specific settings that can be used on multiple contexts, interfaces or gateways.
PSTN	Public Switched Telephone Network. Contains ISDN and POTS
Q.931 Tunneling	Q.931 tunneling is able to support ISDN services and Q.SIG over an IP network.
Q.SIG	ISDN Services comprise additional services for the Private ISDN network such as CNIP (Calling Name Identification Presentation), CNIR (Calling Name Identification Restriction) etc. See also <i>ISDN Services</i> .
Release	A software release describes the main voice and data feature set. It consists of a series of builds.
Routing Engine	The routing engine handles the basic IP routing.
Running Configuration	The currently running configuration (running-config), which is executed from the volatile memory (system:).
Session Router	Calls through Patton device can be routed based on a set of routing criteria. The entity that manages call routing is called Session Router.
Session Initiation Protocol (SIP)	Used for setting up communications sessions (such as conferencing, instant messaging, and telephony) on the Internet
Silence Compression	Silence suppression (or compression) detects the silent periods in a phone conversation and stops the sending of media packets during this periods.
Startup Configuration	The startup configuration is stored in the persistent memory (nvram:) and is always copied for execution to the running configuration in the volatile memory (system:) after a system start-up.
Switching Engine	Part of the Patton device hardware which allows software controlled circuit switching of circuit ports.
System Image	A collective term for application images and interface card driver software, excluding configuration files.
System Memory	The volatile memory, that includes the system: region, holding the running-config for the Trinity during operation of a Patton device.
system:	A region in the volatile memory of a Patton device. See also <i>system memory</i> .

Term or Definition	Meaning
TFTP Server	A central server used for configuration up- and download, download of application and interface card driver software, that is accessed using TFTP.
tftp:	Identification of a remote storing location used for configuration up- and download, download of application and interface card driver software, that is accessed using TFTP.

Appendix B **Command Summary**

Chapter contents

Introduction	930
New Configuration Commands	931
Other	931
Show help	931
Show command history	931
Restart system	931

Introduction

This chapter provides an overview of all CLI commands and modes available. It is organized as follows:

```

Mode Name
Enter Command
Command 1
...
Exit

Mode Name
...

```

Several commands contain a lot of parameters and arguments. The command syntax is described as follows:

- Arguments where you must supply the value are surrounded by <angle brackets>.
- Optional arguments within commands are shown in square brackets ([]).
- Alternative parameters within commands are separated by vertical bars (|).
- Alternative but required parameters are shown within grouped braces ({ }) and are separated by vertical bars (|).

Command syntax is illustrated by an example in [figure 141](#).

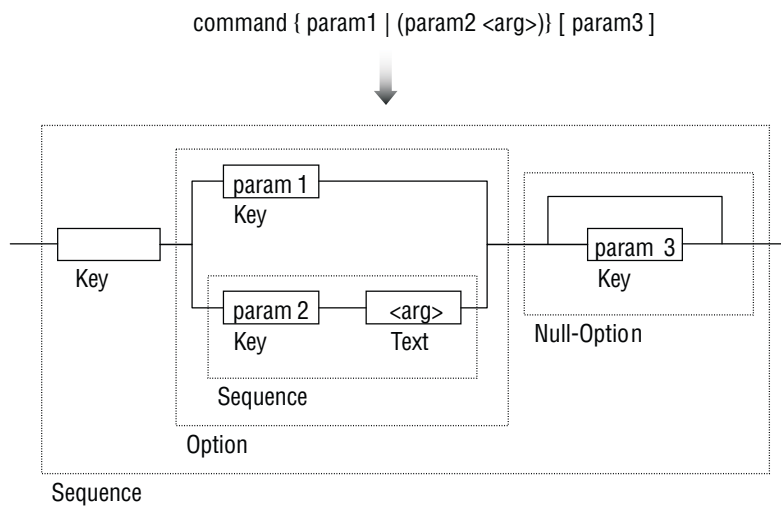


Figure 141. EBNF syntax

New Configuration Commands

The commands documented in the Release Note only cover new additions which are not yet included in the current revision of the Software Configuration Guide. You may download the release notes at www.patton.com/support.

Current Revision:

Document Number: 13211U8-001 Rev. D

Part Number: 07MSWR320_SCG

Revised: July 17, 2006

Other

Show help

Step	Command	Purpose
1	help [<i>topic</i>]	Shows command help.

Show command history

Step	Command	Purpose
1	history	Shows command history.

Use CTRL-N and CTRL-P to browse. The cursor keys (up, down) are not working.

Restart system

Step	Command	Purpose
1	reset	Restarts the system.

Appendix C **Glossary of Terms**

Chapter contents

List of terms	933
---------------------	-----

List of terms

Term	Meaning
Numeric	
10Base-T	Ethernet Physical Medium
A	
AAL	ATM Adaptive Layer
ABR	Available Bit Rate
AC	Alternating Current
AOC	Advice of Charge
ATM	Asynchronous Transfer Mode
audio 3.1	ISDN Audio Service up to 3.1 kHz
audio 7.2	ISDN Audio Service up to 7.2 kHz
B	
BRA	Basic Rate Access
BRI	Basic Rate Interface
C	
CAC	Carrier Access Code
CBR	Constant Bit Rate
CD ROM	Compact Disc Read Only Memory
CDR	Call Detail Record
CFP	Call Forwarding Procedure
CLEC	Competitive Local Exchange Carriers
CLI	Command Line Interface
CLIP	Calling Line Identification Presentation
CO	Central Office
CPE	Customer Premises Equipment
CPU	Central Processor Unit
CRC32	32 bit Cyclic Redundancy Check
D	
DC	Direct Current
DDI	Direct Dialing In number
DHCP	Dynamic Host Configuration Protocol
DLCI	Data Link Connection Identifier
DSL	Digital Subscriber Line
DSLAM	Digital Subscriber Line Access Multiplexor
DSP	Digital Signal Processor
DTMF	Dual Tone Multi-frequency
E	
E1	Transmission Standard at 2.048 Mb/s

Term	Meaning
E-DSS1	ETSI Euro ISDN Standard
EFS	Embedded File System
ET	Exchange Termination
ETH	Ethernet
F	
FAQ	Frequently Asked Questions
FCC	Federal Communication Commission
FR	Frame Relay
G	
G.711	ITU-T Voice encoding standard
G.723	ITU-T Voice compression standard
GUI	Graphic User Interface
GW	Gateway
H	
HFC	Hybrid Fiber Coax
HTTP	HyperText Transport Protocol
HW	Hardware
I	
IAD	Integrated Access Device
ICMP	Internet Control Message Protocol
ILEC	Incumbent Local Exchange Carriers
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ISDN NT	ISDN Network Termination
ISDN S	ISDN S(subscriber Line) Interface
ISDN T	ISDN T(runk Line) Interface
ISDN TE	ISDN Network Terminal Mode
ITC	Information Transfer Bearer Capability
L	
L2TP	Layer Two Tunneling Protocol
LAN	Local Area Network
LCR	Least Cost Routing
LDAP	Lightweight Directory Access Protocol
LE	Local Exchange
LED	Light Emitting Diode
LT	Line Termination
M	
MIB II	Management Information Base II
Modem	Modulator – Demodulator

Term	Meaning
MSN	Multiple Subscriber Number
N	
NAPT	Network Address Port Translation
NAT	Network Address Translation
NIC	Network Interface Card
NT	Network Termination
NT1	Network Termination 1
NT2	Network Termination 2
NT2ab	Network Termination with 2a/b Connections
O	
OEM	Original Equipment Manufacturer
OSF	Open Software Foundation
OSPF	Open Shortest Path First
P	
PBR	Policy Based Routing (principles)
PBX	Private Branch Exchange
PC	Personal Computer
PMC	Production Technology Management Committee
POP	Point of Presence
POTS	Plain Old Telephony Service
PRA	Primary Rate Access
PRI	Primary Rate Interface
PSTN	Public Switched Telephone Network
pt-mpt	point-to-multi point
pt-pt	point-to-point
PVC	Permanent Virtual Circuit
pwd	Password
PWR	Power
Q	
QoS	Quality of Service
R	
RIPv1	Routing Information Protocol Version 1
RIPv2	Routing Information Protocol Version 2
RJ-45	Western Connector Type
RTM	Route Table Manager
RTP	Real-time Protocol
S	
S1	node-connection for Trunk Line
S2	node-connection for Subscriber Line

Term	Meaning
SAR	Segmentation and Reassembly
S-Bus	Subscriber Line (Connection) Bus
SCN	Switched Circuit Network
SCTP	Stream Control Transmission Protocol
SDSL	Symmetric Digital Subscriber Line
SGCP	Simple Gateway Control Protocol
SIP	Session Initiation Protocol.
SME	Small and Medium Enterprises
SNMP	Simple Network Management Protocol
SOHO	Small Office Home Office
SONET	Synchronous Optical Network
SS7	Signaling System No. 7
STM	SDH Transmission at 155 Mb/s
SVC	Switched Virtual Circuit
SW	Software
T	
TCP/IP	Transport Control Protocol/Internet Protocol
TE	Terminal Equipment
TFTP	Trivial File Transfer Protocol
U	
UBR	Unspecified Bit Rate
UD 64	Unrestricted Data 64 kb/s
UDP	User Datagram Protocol
V	
VBR	Variable Bit Rate
VCI	Virtual Channel Identifier
VoIP	Voice over Internet Protocol
VPI	Virtual Path Identifier
W	
WAN	Wide Area Network