

SmartNode Series
SmartWare Release 3.10

Software Configuration Guide

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

Document Number: **13211U4-002 Rev. C**
Part Number: **07MDSNR310SW**
Revised: **September 24, 2004**

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
url: www.patton.com
e-mail: support@patton.com

Copyright Statement

Copyright © 2003 & 2004, Patton Electronics Company. All rights reserved.

Trademark Statement

The terms *SmartWare*, *SmartView*, and *SmartNode* 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.

Contents

Contents	1
Compliance Information	19
Radio and TV Interference	19
CE Notice	19
EU Declaration of Conformity	19
The basis on which conformity is being declared	19
Service	20
About this guide	21
Audience.....	21
How to read this guide	21
Structure.....	21
Typographical conventions used in this document.....	25
General conventions	25
Mouse conventions	26
Service	26
1 System overview	27
Introduction.....	28
SmartNode hardware platforms.....	29
SmartWare embedded software	30
SmartView management center tools.....	31
Applications.....	31
Carrier networks	32
Enterprise networks	32
LAN telephony	34
2 Configuration concepts	35
Introduction.....	36
Contexts and Gateways.....	37
Context	37
Gateway	37
Interfaces, Ports, and Bindings.....	38
Interfaces	38
Ports and circuits	38
Bindings	38
Profiles and Use commands.....	39
Profiles	39
Use Commands	39
3 Command line interface (CLI)	41
Introduction.....	42
Command modes	42

- CLI prompt42
- Navigating the CLI43
 - Initial mode43
 - System changes43
 - Configuration43
 - Changing Modes43
- Command editing43
 - Command help43
 - The No form43
 - Command completion43
 - Command history44
 - Command Editing Shortcuts44
- 4 Accessing the CLI 47**
 - Introduction48
 - Accessing the SmartWare CLI task list.....48
 - Accessing via the console port49
 - Console port procedure49
 - Accessing via a Telnet session50
 - Telnet Procedure51
 - Log onto the SmartWare51
 - Selecting a secure password52
 - Configure operators and administrators52
 - Factory preset administrator account52
 - Creating an operator account52
 - Creating an administrator account53
 - Displaying the CLI version54
 - Displaying account information54
 - Switching to another account54
 - Checking identity and connected users55
 - Ending a Telnet or console port session56
- 5 Establishing basic IP connectivity 57**
 - Introduction58
 - IP context selection and basic interface configuration tasks.....58
 - Entering the IP context, creating IP interfaces and assigning an IP address58
 - Defining IP Ethernet encapsulation and binding an IP interface to a physical port59
 - Activating a physical port59
 - Displaying IP interface information60
 - Deleting IP interfaces61
 - Examples62
 - Setting up an IP interface on an Ethernet port62
- 6 System image handling..... 65**
 - Introduction66
 - Memory regions in SmartWare.....66

Boot procedure	68
Bootloader (for SmartNode 1000 and 2000 Series)	69
Start Bootloader and login	69
Main shell and domains	69
Route Table Manager (RTM)	70
Download Agent	71
Diagnostic	72
Bootloader (for SmartNode 4110/4520 Series)	73
Start Bootloader	73
Start-up with factory configuration	74
Load a new application image (SmartWare) via TFTP	74
Load a new application image (SmartWare) via the serial link	76
Factory configuration	76
System image handling task list	77
Displaying system image information	77
Copying system images from a network server to Flash memory	78
Copying driver software from a network server to Flash memory	79
7 Configuration file handling.....	81
Introduction	82
Understanding configuration files	82
Factory configuration	84
Configuration file handling task list.....	85
Copying configurations within the local memory	86
Replacing the startup configuration with a configuration from Flash memory	87
Copying configurations to and from a remote storage location	89
Replacing the startup configuration with a configuration downloaded from TFTP server	90
Displaying configuration file information	91
Modifying the running configuration at the CLI	91
Modifying the running configuration offline	92
Deleting a specified configuration	93
8 Basic system management	95
Introduction	96
Basic system management configuration task list	96
Managing feature license keys	97
Setting system information	98
Setting the system banner	99
Setting time and date	100
Display clock information	100
Display time since last restart	100
Configuring and starting the Web server	101
Determining and defining the active CLI version	101
Restarting the system	102
Displaying the system logs	102

Controlling command execution	103
Displaying the checksum of a configuration	104
Configuration of terminal sessions	105
9 Radius Client Configuration	107
Introduction	108
The AAA component	108
General AAA Configuration	109
Radius configuration	111
Configuring Radius clients	112
Configuring the Radius server	113
Attributes in the Radius request message	113
Attributes in the Radius accept message	113
Configuring the local database accounts	114
10 IP context overview	117
Introduction	118
IP context overview configuration task list	119
Planning your IP configuration	120
IP interface related information	120
Serial interface related information	121
QoS related information	121
Configuring Ethernet and serial ports	121
Creating and configuring IP interfaces	121
Configuring NAPT	122
Configuring static IP routing	122
Configuring RIP	122
Configuring access control lists	123
Configuring quality of service (QoS)	123
11 IP interface configuration	125
Introduction	126
Software IP interface configuration task list	126
Creating an IP interface	126
Deleting an IP interface	127
Setting the IP address and netmask	127
ICMP message processing	128
ICMP redirect messages	128
Router advertisement broadcast message	129
Defining the MTU and MSS of the interface	129
Configuring an interface as a point-to-point link	130
Displaying IP interface information	131
Testing connections with the ping command	131
Traceroute	132
Examples	132
Deleting an IP interface	132

12 NAT/NAPT configuration	133
Introduction.....	134
Dynamic NAPT	134
Static NAPT	135
Dynamic NAT	136
Static NAT	136
NAPT traversal	137
NAT/NAPT configuration task list	137
Creating a NAPT profile	137
Activate NAT/NAPT	138
Displaying NAT/NAPT configuration information	139
13 Ethernet port configuration	141
Introduction.....	142
Ethernet port configuration task list	142
Entering the Ethernet port configuration mode	143
Configuring medium for an Ethernet port	143
Configuring Ethernet encapsulation type for an Ethernet port	144
Binding an Ethernet port to an IP interface	144
Selecting the frame format for an Ethernet port	145
Configuring layer 2 CoS to service-class mapping for an Ethernet port	146
Adding a receive mapping table entry	147
Adding a transmit mapping table entry	147
Closing an Ethernet port	148
14 Link scheduler configuration	151
Introduction.....	152
Applying scheduling at the bottleneck	152
Using traffic classes	152
Introduction to Scheduling	153
Priority	153
Weighted fair queuing (WFQ)	153
Shaping	153
Burst tolerant shaping or wfq	154
Hierarchy	154
Quick references.....	155
Setting the modem rate	155
Command cross reference	156
Link scheduler configuration task list.....	156
Defining the access control list profile	157
Packet classification	157
Creating an access control list	158
Creating a service policy profile	159
Specifying the handling of traffic-classes	161
Defining fair queuing weight	161

Defining the bit-rate	162
Defining absolute priority	162
Defining the maximum queue length	162
Specifying the type-of-service (TOS) field	162
Specifying the precedence field	163
Specifying differentiated services codepoint (DSCP) marking	163
Specifying layer 2 marking	164
Defining random early detection	165
Discarding Excess Load	165
Devoting the service policy profile to an interface	166
Displaying link arbitration status	167
Displaying link scheduling profile information	167
Enable statistics gathering	167
15 Serial port configuration	169
Introduction	170
Serial port configuration task list	170
Disabling an interface	171
Enabling an interface	171
Configuring the serial encapsulation type	172
Configuring the hardware port protocol	173
Configuring the active clock edge	174
Enter Frame Relay mode	175
Configuring the LMI type	175
Configuring the keep-alive interval	176
Enabling fragmentation	176
Entering Frame Relay PVC configuration mode	178
Configuring the PVC encapsulation type	178
Binding the Frame Relay PVC to IP interface	179
Enabling a Frame Relay PVC	180
Disabling a Frame Relay PVC	181
Debugging Frame Relay	181
Displaying serial port information	182
Displaying Frame Relay information	183
Integrated service access	184
16 Basic IP routing configuration	187
Introduction	188
Routing tables	188
Static routing	188
Basic IP routing configuration task list	188
Configuring static IP routes	188
Deleting static IP routes	189
Displaying IP route information	190
Examples	191

Basic static IP routing example	191
17 RIP configuration.....	193
Introduction	194
Routing protocol	194
RIP configuration task list	195
Enabling send RIP	195
Enabling an interface to receive RIP	196
Specifying the send RIP version	196
Specifying the receive RIP version	197
Enabling RIP learning	197
Enabling an interface to receive RIP	198
Enabling RIP announcing	198
Enabling RIP auto summarization	199
Specifying the default route metric	199
Enabling RIP split-horizon processing	200
Enabling the poison reverse algorithm	200
Enabling holding down aged routes	201
Displaying RIP configuration of an IP interface	201
Displaying global RIP information	202
18 Access control list configuration.....	203
Introduction	204
About access control lists	204
What access lists do	204
Why you should configure access lists	204
When to configure access lists	205
Features of access control lists	205
Access control list configuration task list.....	206
Mapping out the goals of the access control list	206
Creating an access control list profile and enter configuration mode	207
Adding a filter rule to the current access control list profile	207
Adding an ICMP filter rule to the current access control list profile	209
Adding a TCP, UDP or SCTP filter rule to the current access control list profile	211
Binding and unbinding an access control list profile to an IP interface	213
Displaying an access control list profile	214
Debugging an access control list profile	214
Examples	216
Denying a specific subnet	216
19 SNMP configuration	217
Introduction	218
Simple Network Management Protocol (SNMP)	218
SNMP basic components	218
SNMP basic commands	218
SNMP management information base (MIB)	219

Network management framework	219
Identification of the SmartNode 1000, 2000 and 4000 Series via SNMP	220
SNMP tools.....	221
SNMP configuration task list	221
Setting basic system information.....	222
Setting access community information	224
Setting allowed host information	225
specifying the default SNMP trap target	225
Displaying SNMP related information	226
Using the AdventNet SNMP utilities	227
Using the MibBrowser	227
Using the TrapViewer	228
Standard SNMP version 1 traps.....	230
SNMP interface traps	232
20 SNTP client configuration	233
Introduction	234
SNTP client configuration task list	234
Selecting SNTP time servers	235
Defining SNTP client operating mode	235
Defining SNTP local UDP port	236
Enabling and disabling the SNTP client	237
Defining SNTP client poll interval	237
Defining SNTP client constant offset to GMT	237
Defining the SNTP client anycast address	238
Enabling and disabling local clock offset compensation	239
Showing SNTP client related information	240
Debugging SNTP client operation	240
Recommended public SNTP time servers.....	241
NIST Internet time service	241
Other public NTP primary (stratum 1) time servers	242
Additional information on NTP and a list of other NTP servers	243
Recommended RFC	243
21 DHCP configuration.....	245
Introduction.....	246
DHCP-client configuration tasks.....	247
Enable DHCP-client on an IP interface	247
Release or renew a DHCP lease manually (advanced)	248
Get debug output from DHCP-client	249
DHCP-server configuration tasks	250
Configure DHCP-server profiles	250
Use DHCP-server profiles and enable the DHCP-server	252
Check DHCP-server configuration and status	253
Get debug output from the DHCP-server	253

22 DNS configuration	255
Introduction.....	256
DNS configuration task list.....	256
Enabling the DNS resolver.....	256
Enabling the DNS relay.....	257
23 DynDNS configuration	259
Introduction.....	260
DynDNS configuration task list.....	260
Creating a DynDNS account.....	260
Configuring the DNS resolver.....	260
Configuring basic DynDNS settings.....	261
Configuring advanced DynDNS settings (optional).....	261
Defining a mail exchanger for your hostname.....	261
Troubleshooting.....	262
24 PPP configuration	265
Introduction.....	266
PPP configuration task list.....	267
Creating an IP interface for PPP.....	267
Creating a PPP subscriber.....	269
Configuring a PPPoE session.....	270
Configuring a serial port for PPP.....	272
Creating a PPP profile.....	273
Displaying PPP configuration information.....	274
Debugging PPP.....	275
Sample configurations.....	279
PPP over Ethernet (PPPoE).....	279
Without authentication, encapsulation multi, with NAPT.....	279
With authentication, encapsulation PPPoE.....	279
PPP over serial link.....	280
Without authentication, numbered interface.....	280
With authentication, unnumbered interface.....	280
25 VPN configuration	281
Introduction.....	282
Authentication.....	282
Encryption.....	282
Transport and tunnel modes.....	283
Key management.....	283
VPN configuration task list.....	283
Creating an IPsec transformation profile.....	283
Creating an IPsec policy profile.....	284
Creating/modifying an outgoing ACL profile for IPsec.....	286
Configuration of an IP interface and the IP router for IPsec.....	287
Displaying IPsec configuration information.....	287

Debugging IPsec	288
Sample configurations	289
IPsec tunnel, DES encryption	289
SmartNode configuration	289
Cisco router configuration	290
IPsec tunnel, AES encryption at 256 bit key length, AH authentication with HMAC-SHA1-96	290
SmartNode configuration	290
Cisco router configuration	291
IPsec tunnel, 3DES encryption at 192 bit key length, ESP authentication with HMAC-MD5-96	291
SmartNode configuration	291
Cisco router configuration	291
26 CS context overview	293
Introduction	294
CS context configuration task list	295
Planning the CS configuration	295
Configuring general CS settings.....	297
Configuring call routing.....	298
Creating and configuring CS interfaces.....	299
Specify call routing	299
Configuring dial tones.....	300
Configuring voice over IP parameters.....	300
Configuring ISDN ports	301
Configuring FXS ports	301
Configuring an H.323 VoIP connection	301
Configuring a SIP VoIP connection	301
Activating CS context configuration	302
Planning the CS context	305
Configuring general CS settings	306
Configuring call routing	306
Configuring VoIP settings	308
Configuring BRI ports	308
Configuring an H.323 VoIP connection	309
Activating the CS context configuration	309
Showing the running configuration	311
27 CS interface configuration.....	315
Introduction	316
CS interface configuration task list	317
Creating and configuring CS interfaces.....	317
Configuring call routing.....	318
Configuring the interface mapping tables	319
Configuring the precall service tables.....	322
28 ISDN interface configuration.....	325
Introduction	326

ISDN interface configuration task list.....	326
Configuring DTMF dialing (optional)	326
Configuring an alternate PSTN profile (optional)	327
29 FXS interface configuration.....	329
Introduction	330
FXS supplementary services description	330
Call holding	330
Call waiting	330
Additional call offering	331
FXS interface configuration task list	331
Configuring a subscriber number (recommended)	331
Configuring an alternate PSTN profile (optional)	332
Configuring caller-ID presentation (optional)	332
Configuring call holding supplementary service (optional)	333
Configuring call waiting supplementary service (optional)	333
Configuring additional call offering supplementary service (optional)	334
30 FXO interface configuration	335
Introduction	336
FXO services description	337
Creating an FXO interface.....	337
Deleting an FXO interface.....	338
FXO interface configuration task list	339
Configuring an alternate PSTN profile (optional)	339
Configuring when the digits are dialed (optional)	340
Configuring the number of rings to wait before answering the call (optional)	341
Configuring how to detect a call has disconnected (optional)	342
Configuring how to detect an outgoing call is connected (optional)	343
Configuring the destination of the call	344
FXO interface examples	345
31 H.323 interface configuration	347
Introduction	348
H.323 interface configuration task list.....	348
Binding the interface to an H.323 gateway	349
Configuring an alternate VoIP profile (optional)	350
Configuring CLIP/CLIR support (optional)	351
Enabling the early call disconnect (optional)	352
Enabling the via address support (optional)	353
Override the default destination call signaling port (Optional)	353
Configuring status inquiry settings (optional)	354
32 SIP interface configuration	355
Introduction	356
SIP interface configuration task list.....	356

- Binding the interface to a SIP gateway357
- Configure a remote host357
- Configuring an alternate VoIP profile (Optional)358
- Configuring early call connect / disconnect (optional)358
- Configuring a phone context (optional)359
- 33 Call router configuration..... 361**
- Introduction363
- Call router configuration task list.....365
 - Map out the goals for the call router365
 - Enable advanced call routing on circuit interfaces366
 - Configure general call router behavior366
 - Configure address completion timeout366
 - Configure default digit collection timeout and terminating character367
 - Configure number prefix for ISDN number types368
 - Configure call routing tables369
 - Create a routing table370
 - Called party number routing table372
 - Regular Expressions372
 - Digit Collection374
 - Digit Collection Variants375
 - Calling party number routing table378
 - Number type routing table379
 - Numbering plan routing table380
 - Name routing table380
 - IP address routing table381
 - URI routing table381
 - Presentation Indicator Routing Table381
 - Screening Indicator Routing Table382
 - Information transfer capability routing table383
 - Time of day routing table384
 - Day of Week Routing Table384
 - Date routing table385
 - Deleting routing tables385
 - Configure mapping tables386
 - E.164 to E.164 Mapping Tables391
 - Other mapping tables393
 - Deleting mapping tables394
 - Creating complex functions395
 - Deleting complex functions395
 - Creating call services396
 - Creating a hunt group service397
 - Creating a distribution group service406
 - Deleting call services408

Activate the call router configuration	408
Test the call router configuration	409
34 Tone configuration.....	417
Introduction	418
Tone-set profiles.....	418
MGCP-Events	419
Tone configuration task list	419
Configuring call-progress-tone profiles	420
Configure tone-set profiles	421
Enable tone-set profile	421
Show call-progress-tone and tone-set profiles	422
35 ISDN port configuration.....	425
Introduction	426
ISDN reference points	426
Possible SmartNode port configurations	427
ISDN UNI Signaling	427
SmartNode 1000 Series	429
IC-4BRV Interface Card	429
ISDN Configuration Concept.....	429
ISDN Layering	429
Configuration example	430
Description	430
ISDN port configuration task list	431
Shutdown and enable ISDN ports	431
Configure BRI port parameters (Layer 1)	432
Configure PRI Port Parameters (Layer 1)	433
Configure ISDN layer 2 parameters (Q921)	434
Configure ISDN layer 3 parameters (Q931)	435
Show ISDN port status	437
Examples	439
36 FXS port configuration	441
Introduction.....	442
Shutdown and enable FXS ports.....	442
Bind FXS ports to higher layer applications.....	443
Configure country-specific FXS port parameters.....	443
Other FXS port parameters.....	444
Choose a low-bit-rate codec for FXS ports.....	445
Example	446
37 FXO port configuration	447
Introduction.....	448
Shutdown and enable FXO ports.....	448
Bind FXO ports to higher layer applications.....	449

Configure country specific FXO port parameters.....	449
Other FXO port parameters	450
38 H.323 gateway configuration	451
Introduction	452
Gateway configuration task list	452
Configure datapath related settings	453
Binding the gateway to an IP interface	454
Enable the gateway	454
Configure registration authentication service (RAS) (Optional)	455
Configure H.235 Security (optional)	456
Advanced configuration options (optional)	460
Enabling H.245 Tunneling	460
Enabling the fastconnect procedure	460
Enabling the early H.245 procedure	461
Changing the TCP port for inbound call-signaling connections	461
Setting the response timeout	461
Setting the connect timeout	462
Configuring the terminal type for registration with the gatekeeper	462
Troubleshooting	463
39 SIP gateway configuration.....	465
Introduction	466
Gateway configuration task list	466
Configure DNS resolver	467
Configure datapath related settings	467
Binding the gateway to an IP interface	468
Enable the Gateway	468
Registering with a registrar (optional)	468
Configure a domain name (optional)	469
Configure a default server (optional)	470
Configure authentication parameters (optional)	471
Enable the session timer (optional)	471
Advanced configuration options (optional)	472
Changing the listening port for inbound call-signaling	472
Define session timer version	472
Define call transfer version	473
Troubleshooting	473
40 VoIP profile configuration	475
Introduction	476
VoIP profile configuration task list	476
Creating a VoIP profile	477
Configure codecs	478
Configuring DTMF relay	480
Configuring RTP payload types	480

Configuring the dejitter buffer (advanced)	480
Enabling/disabling filters (advanced)	483
Configuring Fax transmission	484
Configuring modem transmission	486
Examples	487
Home office in an enterprise network	487
Home office with fax	489
Soft phone client gateway	490
41 PSTN profile configuration.....	493
Introduction	494
PSTN profile configuration task list	494
Creating a PSTN profile	494
Configuring the echo canceller	495
Configuring output gain	495
42 VoIP debugging.....	497
Introduction	498
Debugging strategy	498
Verifying IP connectivity	498
Debugging call signaling.....	499
Debugging ISDN signaling	500
Verify an incoming call	500
Verify an outgoing call	502
Verify ISDN layer 1,2,3 status	503
Debugging FXS Signaling	504
Verify an incoming call	504
Verify an outgoing call	505
Debugging H.323 Signaling	506
Verify an incoming call	507
Verify an outgoing call	508
Debugging SIP signaling	510
Verify an incoming call	511
Verify an outgoing call	511
Using SmartWare's internal call generator	512
Debugging voice data	513
Check system logs	515
How to submit trouble reports to Patton	515
A Terms and definitions	517
Introduction	518
SmartWare architecture terms and definitions	518
B Mode summary	525
Introduction	526
C Command summary	529

Introduction	531
operator_exec	531
administrator_exec	533
configure	535
Contexts and interfaces	536
context_ip	536
interface	536
dyndns	537
context_cs	537
cr_table_routing	537
cr_table_mapping	537
cr_table_precall-service	537
cr_table_complex_function	538
interface_h323	538
interface_sip	538
interface_isdn	539
interface_fxs	539
interface_fxo	539
service_hunt	539
service_distribute	540
service_second-dialtone	540
Gateways	540
gateway_h323	540
gateway_sip	540
Ports	541
port_ethernet	541
pppoe	541
port_serial	541
framerelay	542
pvc	542
port_virtual	542
port_fxs	542
port_fxo	542
port_isdn	543
port_isdn_q921	543
port_isdn_q931	543
Profiles	543
profile_acl	543
profile_service-policy	544
source	544
profile_napt	544
profile_ppp	545
profile-ipsec-transform	545
ipsec-manual-policy	545

profile_call-progress-tone	545
profile_tone-set	545
profile_voip	546
profile_pstn	548
profile_dhcp-server	548
profile_authentication	548
Other.....	548
radius-client	548
system	548
ic_voice	548
subscriber_ppp	548
Show help	549
Show command history	549
Show RedBoot version	550
Restart system	550
Display memory content	550
Set IP addresses	550
Check network connection to remote system	551
Load a program to memory, so that it can be executed or stored in the Flash memory	551
Execute a program loaded into memory	552
Manage program images in Flash memory	552
Display images stored in Flash memory	552
Load an image into RAM so that it can be started	553
Re-initialize Flash image store	553
Create a new image in the Flash image store	554
Delete an image from the Flash image store	554
RedBoot Configuration	554
Displaying current configuration	554
Modify configuration	555
Re-initialize configuration to default values	555
Read data from EEPROM	555
Enable/Disable cache	555
D Internetworking terms & acronyms	557
Abbreviations.....	558
E Used IP ports & available voice codecs in the SmartWare	563
Used IP ports in the SmartWare.....	564
Available voice codecs in the SmartWare	565

Compliance Information

Radio and TV Interference

The SmartNode series of products generate and use radio frequency energy, and if not installed and used properly—i.e. in strict accordance with the manufacturer's instructions—may cause interference to radio and television reception. The SmartNode devices have been tested and found to comply with the limits for a Class A computing device in accordance with specifications in Subpart B of Part 15 of FCC rules, which are designed to provide reasonable protection from such interference in a commercial installation. However, there is no guarantee that interference will not occur in a particular installation. If a SmartNode series device does cause interference to radio or television reception, which can be determined by disconnecting the unit, the user is encouraged to try to correct the interference by one or more of the following measures: moving the computing equipment away from the receiver, re-orienting the receiving antenna and/or plugging the receiving equipment into a different AC outlet (such that the computing equipment and receiver are on different branches).

CE Notice

The CE symbol on your Patton Electronics equipment indicates that it is in compliance with the Electromagnetic Compatibility (EMC) directive and the Low Voltage Directive (LVD) of the European Union (EU). A Certificate of Compliance is available by contacting Technical Support.

EU Declaration of Conformity

EU Directives covered by this declaration

89/336/EEC	Electromagnetic Compatibility Directive amended by 92/31/EEC & 93/68/EEC
72/23/EEC	Low Voltage Equipment Directive amended by 93/68/EEC

Note During the transition period, products may not comply with the Low Voltage Directive.

The basis on which conformity is being declared

The products identified above comply with the requirements of the above EU directives by meeting the following standards:

- Safety compliance: EN 60950
- EMC compliance: EN 55022, EN 55024
- ETSI TBR3 (BRI)
- TBR4 (PRI)

The CE mark was first applied in 2000.

Service

All warranty and non-warranty repairs must be returned freight prepaid and insured to Patton Electronics. All returns must have a Return Materials Authorization number on the outside of the shipping container. This number may be obtained from Patton Electronics Technical Services at:

- Tel: **+1 (301) 975-1007**
- Email: **support@patton.com**
- URL: **<http://www.patton.com>**

Note Packages received without an RMA number will not be accepted.

About this guide

The objective of this *SmartWare Command Configuration Guide* is to provide information concerning the syntax and usage of the command set. For hardware configuration information, refer to the getting started guide that came with your SmartNode systems.

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 25)

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 SmartNode.
- System administrators with a basic networking background and experience, but who might not be familiar with the SmartNode.
- Operators
- Installers
- Maintenance technicians

How to read this guide

SmartWare is a complex and multifaceted operating system running on your SmartNode device. 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 SmartWare and the philosophy of contexts used for IP and circuit switching related configuration.

- Appendix A, [“Terms and definitions”](#) on page 517 contains the terms and their definitions that are used throughout this *SmartWare Software Configuration Guide*
- Chapter 1, [“System overview”](#) on page 27 provides an overview of the main elements of a SmartNode system.
- Chapter 10, [“IP context overview”](#) on page 117
- Chapter 26, [“CS context overview”](#) on page 293

Structure

This guide contains the following chapters and appendices:

- Chapter 1, [“System overview”](#) on page 27 provides an overview of the main elements of a SmartNode system.
- Chapter 2, [“Configuration concepts”](#) on page 35 introduces basic SmartWare configuration concepts.

- Chapter 3, "[Command line interface \(CLI\)](#)" on page 41 gives an overview of the CLI and the basic features that allow you to navigate the CLI and edit commands effectively.
- Chapter 4, "[Accessing the CLI](#)" on page 47 describes the procedures for entering SmartWare commands via the command line interface (CLI), to obtain help, to change operator mode and to terminate a session.
- Chapter 5, "[Establishing basic IP connectivity](#)" on page 57 explains how to establish network-based connections to and from your SmartNode by using IP interfaces and Ethernet ports.
- Chapter 6, "[System image handling](#)" on page 65 describes how to load and maintain system images and driver software.
- Chapter 7, "[Configuration file handling](#)" on page 81 describes how to upload and download configuration files from and to a SmartNode 1000, 2000, or 4000 Series devices.
- Chapter 8, "[Basic system management](#)" on page 95 describes parameters that report basic system information to the operator or administrator, and their configuration.
- Chapter 9, "[Radius Client Configuration](#)" on page 107 provides an overview of the authentication, authorization, and accounting (AAA) component in SmartWare and describes how to configure the Radius client, a subpart of the AAA component.
- Chapter 10, "[IP context overview](#)" on page 117 outlines SmartWare *Internet protocol* (IP) context, together with its related components.
- Chapter 11, "[IP interface configuration](#)" on page 125 provides a general overview of SmartNode interfaces and describes the tasks involved in their configuration.
- Chapter 12, "[NAT/NAPT configuration](#)" on page 133 provides a general overview of the network address port translation and describes the tasks involved in its configuration.
- Chapter 13, "[Ethernet port configuration](#)" on page 141 provides an overview of Ethernet ports and describes the tasks involved in their configuration through SmartWare.
- Chapter 14, "[Link scheduler configuration](#)" on page 151 describes how to use and configure SmartWare *quality of service* (QoS) features.
- Chapter 15, "[Serial port configuration](#)" on page 169 provides an overview of the serial port and describes the tasks involved in its configuration through SmartWare.
- Chapter 16, "[Basic IP routing configuration](#)" on page 187 provides an overview of IP routing and describes the tasks involved in configuring static IP routing in SmartWare.
- Chapter 17, "[RIP configuration](#)" on page 193 provides an overview of the *routing information protocol* (RIP) and describes the tasks involved in configuring RIP features within SmartWare.
- Chapter 18, "[Access control list configuration](#)" on page 203 provides an overview of IP access control lists and describes the tasks involved in their configuration through SmartWare.
- Chapter 19, "[SNMP configuration](#)" on page 217 provides overview information about the *simple network management protocol* (SNMP) and describes the tasks used to configure those of its features supported by SmartWare.
- Chapter 20, "[SNTP client configuration](#)" on page 233 describes how to configure a *simple network time protocol* (SNTP) client.

- Chapter 21, "[DHCP configuration](#)" on page 245 provides an overview of the *dynamic host configuration control protocol* (DHCP) and describes the tasks involved in its configuration.
- Chapter 22, "[DNS configuration](#)" on page 255 describes how to configure the *domain name system* (DNS) component.
- Chapter 23, "[DynDNS configuration](#)" on page 259 describes configuring the *dynamic DNS* (DynDNS) service.
- Chapter 24, "[PPP configuration](#)" on page 265 describes how to configure the *point-to-point protocol* over different link layers.
- Chapter 25, "[VPN configuration](#)" on page 281 describes how to configure the VPN connections between two SmartNodes or between a SmartNode and a third-party device.
- Chapter 26, "[CS context overview](#)" on page 293 gives an overview of SmartWare *circuit-switching* (CS) context and its associated components and describes the tasks involved in its configuration.
- Chapter 27, "[CS interface configuration](#)" on page 315 gives an overview of interfaces in the CS context and describes the tasks involved in its configuration.
- Chapter 28, "[ISDN interface configuration](#)" on page 325 provides an overview of ISDN interfaces, and the tasks involved in their configuration.
- Chapter 29, "[FXS interface configuration](#)" on page 329 provides an overview of FXS interfaces, and the tasks involved in their configuration.
- Chapter 31, "[H.323 interface configuration](#)" on page 347 provides an overview of H.323 interfaces used by H.323 gateways and describes the specific tasks involved in their configuration.
- Chapter 32, "[SIP interface configuration](#)" on page 355 provides an overview of SIP interfaces used by SIP gateways and describes the specific tasks involved in their configuration.
- Chapter 33, "[Call router configuration](#)" on page 361 provides an overview of call router tables, mapping tables and call services and describes the tasks involved in configuring the call router in SmartWare.
- Chapter 34, "[Tone configuration](#)" on page 417 gives an overview of SmartWare call-progress-tone profiles and tone-set profiles and describes the tasks involved in their configuration.
- Chapter 35, "[ISDN port configuration](#)" on page 425 provides an overview of SmartNode ISDN ports and describes the tasks involved in configuring ISDN ports in SmartWare.
- Chapter 36, "[FXS port configuration](#)" on page 441 provides an overview of POTS signaling and SmartNode FXS ports and describes the tasks involved in configuring FXS ports in SmartWare.
- Chapter 38, "[H.323 gateway configuration](#)" on page 451 provides an overview of the H.323 gateway and describes the tasks involved in its configuration.
- Chapter 39, "[SIP gateway configuration](#)" on page 465 provides an overview of the SIP gateway and describes the tasks involved in its configuration.
- Chapter 40, "[VoIP profile configuration](#)" on page 475 gives an overview of SmartWare VoIP profiles, how they are used and describes the tasks involved in VoIP profile configuration.
- Chapter 41, "[PSTN profile configuration](#)" on page 493 gives an overview of SmartWare PSTN profiles, and describes how they are used and the tasks involved in PSTN profile configuration.

- Chapter 42, "[VoIP debugging](#)" on page 497 helps you to localize a system component that is responsible for faults during operation of a SmartNode device.
- Appendix A, "[Terms and definitions](#)" on page 517 contains the terms and their definitions that are used throughout this *SmartWare Software Configuration Guide*.
- Appendix B, "[Mode summary](#)" on page 525 illustrates the modes hierarchy.
- Appendix C, "[Command summary](#)" on page 529 is a command reference.
- Appendix D, "[Internetworking terms & acronyms](#)" on page 557 contains terms and definitions relating to internetworking.
- Appendix E, "[Used IP ports & available voice codecs in the SmartWare](#)" on page 563 describes the used IP ports and available voice codecs in SmartWare.


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.
Futura bold type	Commands and keywords are in boldface font.
<i>Futura bold-italic type</i>	Parts of commands, which are related to elements already named by the user, are in boldface italic font.
<i>Italicized Futura type</i>	Variables for which you supply values are in <i>italic</i> font
Futura 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 ()
blue screen	Information you enter is in blue screen font.
screen	Terminal sessions and information the system displays are in screen font.
node	The leading IP address or nodename of a SmartNode is substituted with node in boldface italic font.
SN	The leading SN on a command line represents the nodename of the Smart-Node
#	An hash sign at the beginning of a line indicates a comment line.

Mouse conventions

The following conventions are used when describing mouse actions:

Table 2. Mouse conventions

Convention	Meaning
Left mouse button	This button refers to the primary or leftmost mouse button (unless you have changed the default configuration).
Right mouse button	This button refers the secondary or rightmost mouse button (unless you have changed the default configuration).
Point	This word means to move the mouse in such a way that the tip of the pointing arrow on the screen ends up resting at the desired location.
Click	Means to quickly press and release the left or right mouse button (as instructed in the procedure). Make sure you do not move the mouse pointer while clicking a mouse button.
Double-click	Means to press and release the same mouse button two times quickly
Drag	This word means to point the arrow and then hold down the left or right mouse button (as instructed in the procedure) as you move the mouse to a new location. When you have moved the mouse pointer to the desired location, you can release the mouse button.

Service

All warranty and non-warranty repairs must be returned freight prepaid and insured to Patton Electronics. All returns must have a Return Materials Authorization number on the outside of the shipping container. This number may be obtained from Patton Electronics Technical Services at:

- Tel: **+1 (301) 975-1007**
- Email: **support@patton.com**
- URL: **http://www.patton.com**

Note Packages received without an RMA number will not be accepted.

Chapter 1 **System overview**

Chapter contents

Introduction	28
SmartNode hardware platforms	29
SmartWare embedded software	30
SmartView management center tools	31
Applications.....	31
Carrier networks	32
Enterprise networks	32
LAN telephony	34

Introduction

This chapter provides an overview of the main elements of a SmartNode system and includes the following sections:

- SmartNode hardware platforms (see [page 29](#))
- SmartWare embedded software (see [page 30](#))
- SmartView Management Center tools (see [page 31](#))

A complete SmartNode system or network, as installed in any of the application scenarios introduced in section “[Applications](#)” on page 31, is typically composed of the following main elements plus a third-party network infrastructure:

- The first and most obvious element is the *SmartNode* devices (also referred to as *hardware platforms* or *network nodes*) that provide the physical connectivity, the CPU and DSP resources. All SmartNode models support packet-routed and circuit-switched traffic equally well.
- The second element comprises the embedded software—called *SmartWare*—running on the SmartNode hardware platforms.
- The third element is the set of *SmartView Management Center* tools for configuring and controlling SmartWare and SmartNodes in a network. Complementing the management interfaces provided in SmartWare, the SmartView tools enable network administrators and operators to handle large numbers of SmartNode devices.
- 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 services nodes.

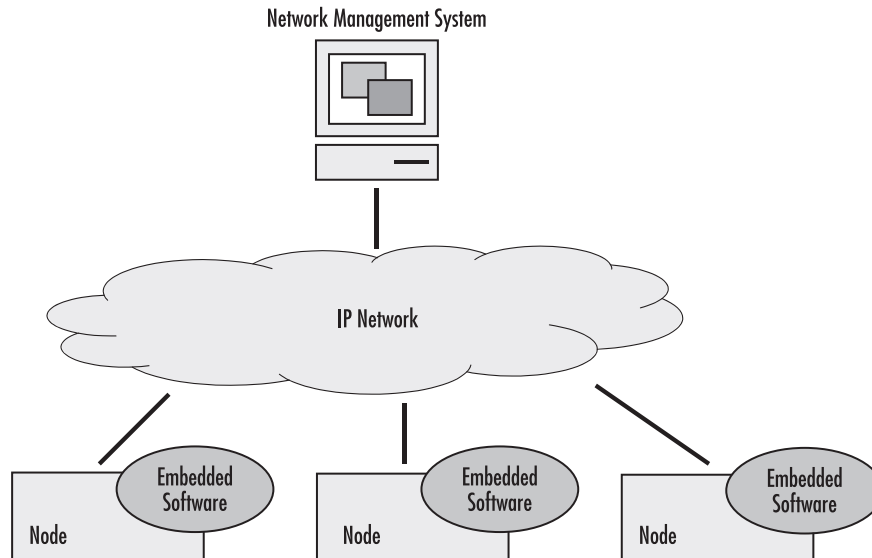


Figure 1. System overview with a SmartView Management Center

SmartNode hardware platforms

The SmartNode series of devices covers a performance range varying from that suitable for small office/home office (SOHO) applications to large corporate sites, or in terms of voice channels from 2 channels (one BRI/So or 2 FXS) to 120 (two PRI/S2m). The SmartNodes comprise the following classes:

- The SmartNode 1000 series compact devices with fixed configured on-board BRI/So ports
- The SmartNode 2000 series with on-board ports plus expansion slots for individual interface configurations using a range of optional interface cards (IC).
- The SmartNode 4000 series compact devices with fixed on-board analog ports

Figure 2 depicts the basic system model of a Patton SmartNode. All SmartNode 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. H.323-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.

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

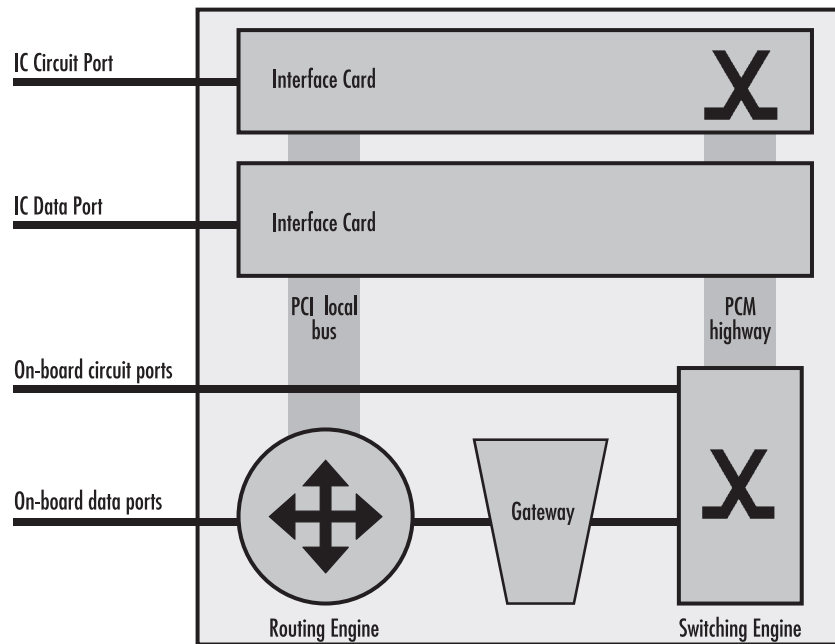


Figure 2. SmartNode System Model

SmartWare embedded software

SmartWare is the application software that runs on the SmartNode hardware platforms. SmartWare is available in several releases that support all available SmartNode models. Refer to SmartWare release notes for detailed information about hardware support.

For each SmartWare release there are platform-specific *build* numbers. There may be more than one build per release and platform as updates become available. Refer to SmartWare release notes for build numbers and build-specific enhancements and limitations.

A SmartWare build is a binary image file. It is usually divided into several checksum-protected files to improve download efficiency and security. The download to the SmartNode is handled in sequence by using a download *batchfile*. Refer to chapter 6, “[System image handling](#)” on page 65 for details on SmartWare image downloads.

In addition to the actual SmartWare images there are several additional embedded software components that you will encounter:

- The *boot loader* is a “mini” application that performs basic system checks and starts SmartWare application. It also provides minimal network services, allowing the SmartNode to be accessed and upgraded over the network even if SmartWare application should not start. The boot loader is installed in the factory and requires no upgrading.
- The *PMC loader* initializes the PMC interface cards when mounted in SmartNode 2000 series devices. It checks the hardware versions and determines whether compatible PMC drivers are available. The PMC loader may be upgraded together with a SmartWare release.
- The *PMC driver* software performs the runtime tasks on the PMC interface cards mounted in SmartNode 2000 series devices. The PMC drivers are interface card specific and also have build numbers. Refer to the

SmartWare release notes for PMC driver software compatibility. The PMC drivers may be upgraded together with a SmartWare release or they can be downloaded individually onto the device's flash memory file system.

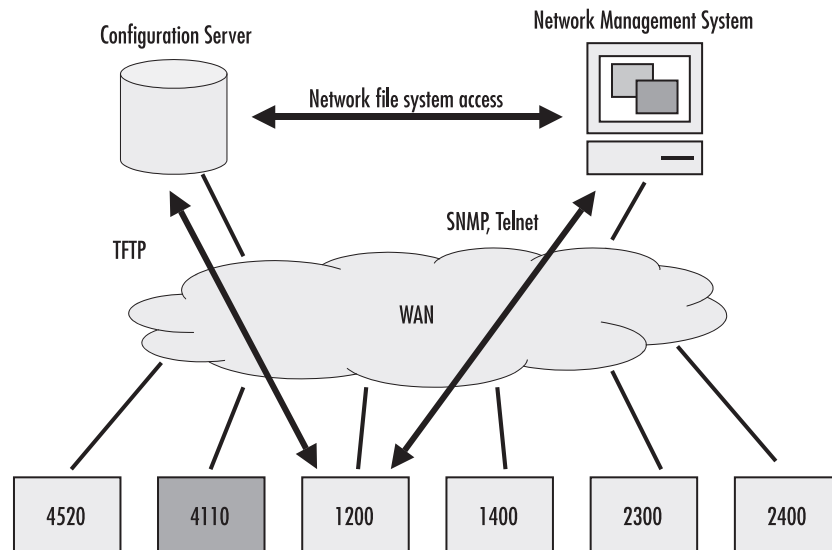


Figure 3. SmartNode Management System

SmartView management center tools

SmartWare provides two management interfaces:

- The *Command Line Interface (CLI)*, which supports full online configuration and monitoring access for the operator
- The *SNMP agent* and *MIB*, with an emphasis on inventory and alarm management for integration in a third-party *Network Management System (NMS)*

With the aid of configuration files and TFTP up and downloads, the SmartNodes can also be managed offline using standard text editors and file systems.

A number of host-based management applications are available to facilitate generating, editing, and maintaining configuration files. Tools are also available for integrating SmartNode management into standard network management platforms such as HP OpenView.

Applications

The Patton SmartNode 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 SmartNode applications and the main elements in a SmartNode network.

- **Carrier networks**—SmartNodes are used as customer gateways or integrated access devices at the customer premises. These applications are also called Integrated Service Access (ISA).
- **Enterprise networks**—SmartNodes are used as WAN routers and voice gateways for inter-site networking. These applications are also called *multiservice intranets (MSI)*.

- **LAN telephony**—SmartNodes 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.

SmartNode 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.

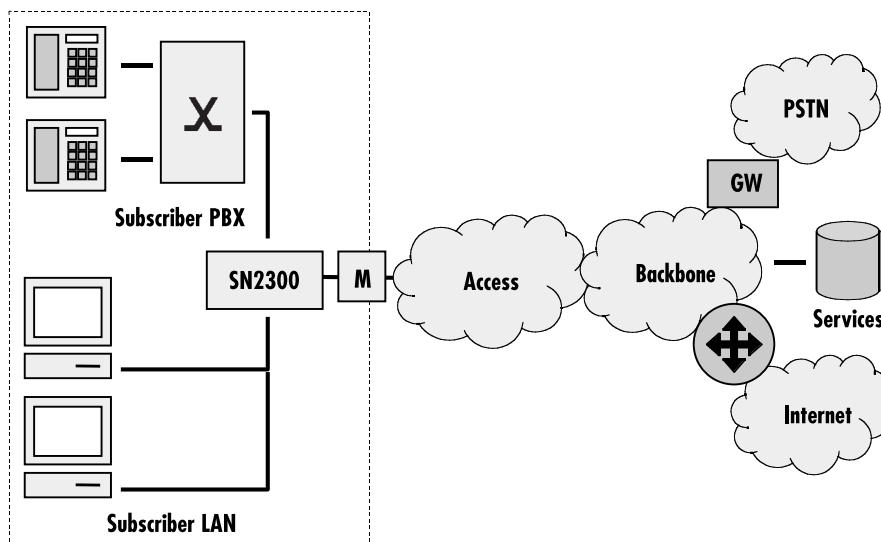


Figure 4. Typical carrier network application with a SmartNode 2300

Figure 4 shows the deployment of SmartNodes in carrier networks. Each subscriber site is equipped with a SmartNode that connects the subscriber CPE 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, and conventional leased lines. With the use of an external modem, the SmartNode can connect to leased lines or any bridged-Ethernet broadband access.

Enterprise networks

In company-owned and operated wide area networks, SmartNodes 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, the SmartNodes provide migration and investment protection for legacy telephony systems.

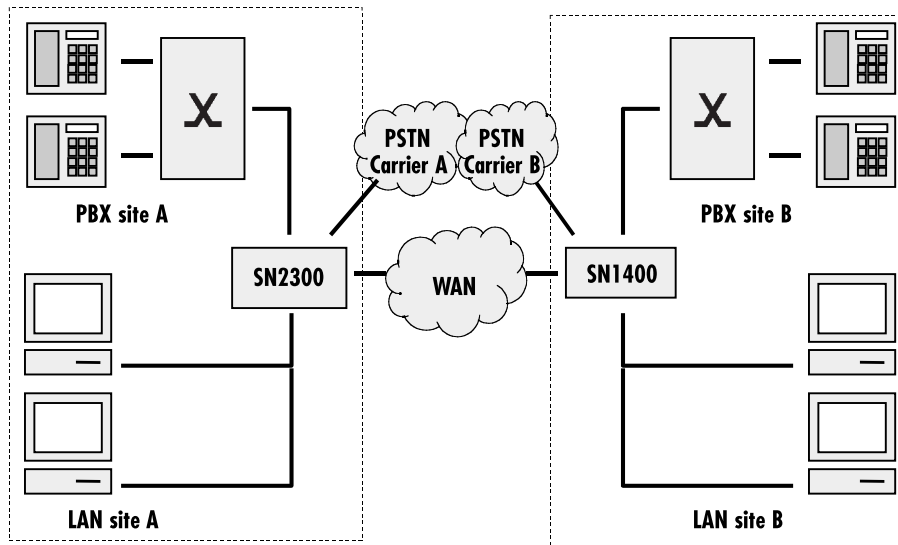


Figure 5. Typical enterprise network with a SmartNode 1400 and 2300

Figure 5 shows the deployment of SmartNodes in enterprise networks. Each site (headquarter, branch or home office) is equipped with a SmartNode that connects the local LAN and telephony infrastructure with the IP WAN and the local PSTN carrier.

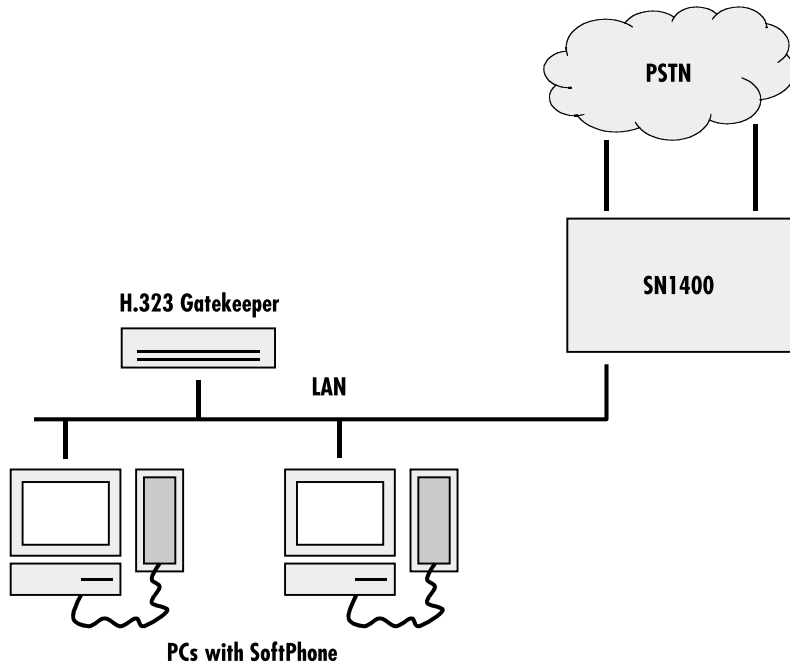


Figure 6. Typical LAN telephony system with a SmartNode 1400 gateway

LAN telephony

With its voice-over-IP gateway features, the SmartNode can be used as a standalone gateway for H.323 or SIP LAN voice systems such as LAN-based PBXs or call centers (see [figure 6](#)).

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

[Figure 6](#) shows the deployment of a SmartNode 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.

Chapter 2 **Configuration concepts**

Chapter contents

Introduction	36
Contexts and Gateways.....	37
Context	37
Gateway	37
Interfaces, Ports, and Bindings.....	38
Interfaces	38
Ports and circuits	38
Bindings	38
Profiles and Use commands.....	39
Profiles	39
Use Commands	39

Introduction

This chapter introduces basic SmartWare 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 37](#))
- Interfaces, ports, and bindings (see [page 38](#))
- Profiles and Use commands (see [page 39](#))

Patton SmartNodes 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, SmartWare configuration is based on a number of abstract concepts that represent the various SmartWare components.

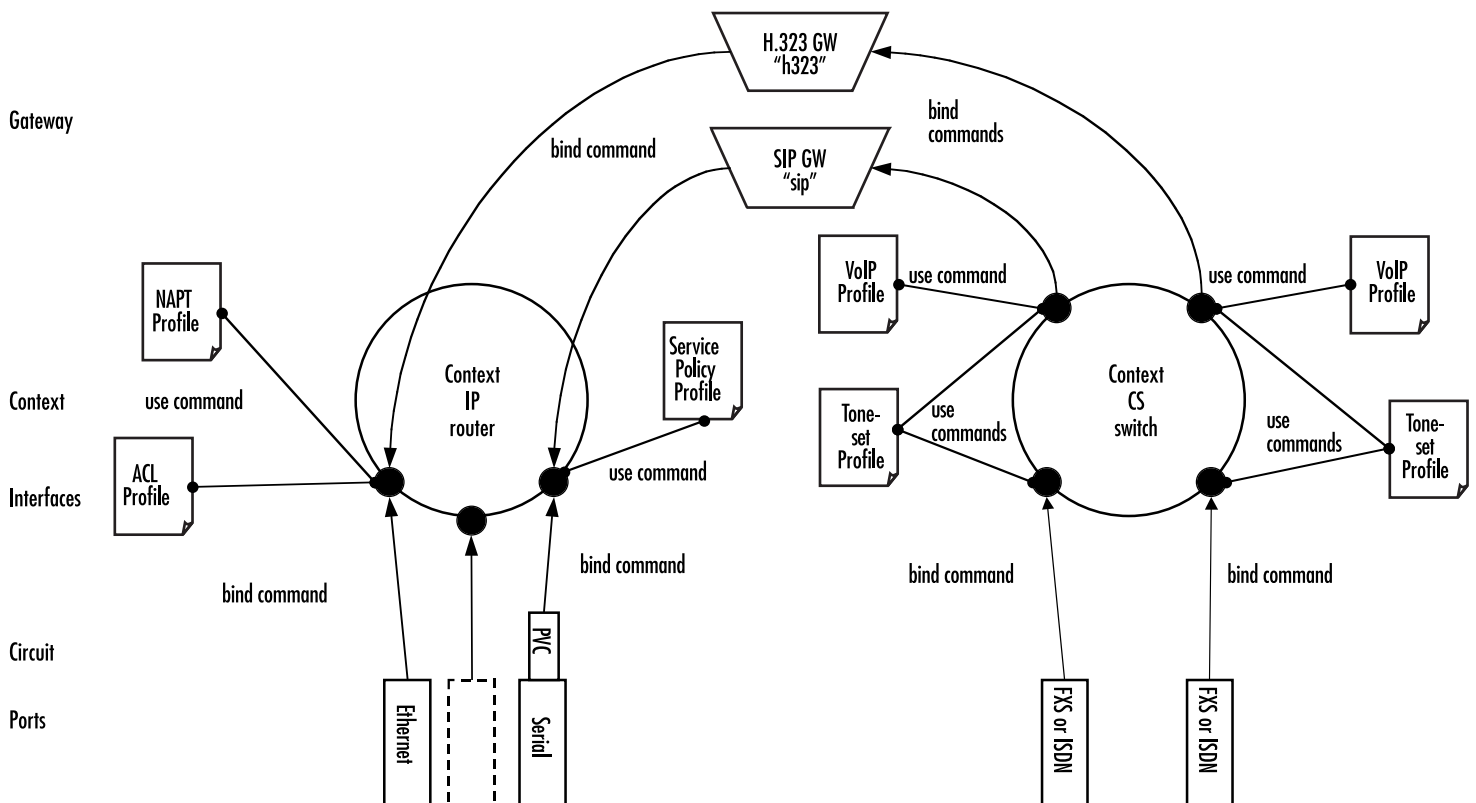


Figure 7. Configuration concept overview

Figure 7 shows the various elements of a complete SmartNode 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 7](#) on page 36 and describe the concepts and elements in more detail.

Contexts and Gateways

Context

A SmartWare *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 SmartNode. For example:

- A CS context contains the circuit-switching functions of the SmartNode. It can be thought of as an embedded multiplexer or cross-connect within the SmartNode
- An IP context contains the routing functions of the SmartNode. It can be thought of as an embedded router within the SmartNode

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 bridging, ATM, or FR switching becomes available so a bridging, ATM, or FR context can be introduced.

Each context contains a number of *interfaces*, which build the connections to other SmartWare elements and the outside world. [Figure 7](#) on page 36 shows two contexts:

- one of type IP named *router*
- one of type CS named *switch*

This corresponds to the default configuration of all SmartNodes.

Note SmartWare 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, an H.323-gateway can connect 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, a separate gateway can be built for newly-supported technology such as MGCP or SIP without complicating the configuration methods of existing software parts. [Figure 7](#) on page 36 shows two gateways, one of type h323 named *h323* and one of type SIP named *sip*.

Example

An H.323 gateway named *h323-gw* has an H.323 gateway ID and an associated gatekeeper configuration. It is connected to the interface *ip-trunk* on the circuit-switch context *switch* and the interface *global-wan* on the IP context *router*.

Interfaces, Ports, and Bindings

Interfaces

The concept of an interface in SmartWare differs from that in traditional networking devices. Traditionally, the term *interface* is often synonymous with *port* or *circuit*, which are physical entities. In SmartWare 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 SmartWare.

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 SmartWare. Refer to the “Bindings” section for more information. In [figure 7](#) on page 36, 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 SmartWare represent the physical connectors and channels on the SmartNode 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 SmartNode ports are: 10Base-T Ethernet, Serial ISDN BRI, and ISDN PRI, analog FXS and FXO. Ports are numbered according to the SmartNode port numbering scheme. The port name corresponds to the label (or abbreviation) printed on the hardware.

Example: Ethernet 0/1, Serial 0/0, BRI 3/2

Some ports may contain multiple *circuits*. For example, serial ports can contain one or more Frame Relay Permanent Virtual Circuits (PVC). If a port has one or more circuits configured, the individual circuits are bound to *interfaces* on a context. The port itself may not be bound in that case.

Example: frame-relay pvc 112.

[Figure 7](#) on page 36 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.

In the case of IP interfaces, 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 PSTN CS interfaces (BRI, PRI, FXS, FXO interfaces), 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 (H.323, SIP), bindings are configured statically in the CS interface configuration. The binding is created from the interface to the gateway.

Bindings from ports to IP interfaces and from CS interfaces to ISDN ports are shown in [figure 7](#) on page 36.

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 7](#) on page 36.

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	42
Command modes	42
CLI prompt	42
Navigating the CLI	43
Initial mode	43
System changes	43
Configuration	43
Changing Modes	43
Command editing	43
Command help	43
The No form	43
Command completion	43
Command history	44
Command Editing Shortcuts	44

Introduction

The primary user interface to SmartWare is the command line interface (CLI). You can access the CLI via the SmartNode console port or through a Telnet session. The CLI lets you configure the complete SmartWare functionality, as opposed to the SNMP and HTTP management interfaces that offer a more limited subset of the functions. 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 43](#))

Command modes

The CLI is composed of modes. There are two *mode groups*: the *exec mode* group and the *configuration mode* group. Within the exec mode group there are two modes: *operator exec* and *administrator exec*. 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. Appendix B, “[Mode summary](#)” on page 525 contains a detailed overview of all command modes, and appendix C, “[Command summary](#)” on page 529 describes the commands that are valid in each mode.

CLI prompt

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

```
nodename>
```

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

```
nodename#
```

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

```
nodename(mode)[name]#
```

Where:

- *nodename* is the currently configured name of the SmartNode, 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

Example: the prompt in **radius-client mode**, assuming the nodename *SN* and the instance *deepblue* is:

```
SN(radius)[deepblue]#
```

The CLI commands used to enter each mode and the system prompt that is displayed when you are working in each mode is summarized in appendix B, “[Mode summary](#)” on page 525.

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. After doing that, other configuration modes are accessible, as diagrammed in the overview in [figure 7](#) on page 36.

Changing Modes

The **exit** command moves the user up one level in the mode hierarchy (the same command works in any of configuration modes). For example, when in *pvc* configuration mode, typing **exit** will take you to *framerelay* configuration mode.

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 Session Router trace tool, enter the command **debug session-router**. To subsequently disable the Session Router trace, enter the command **no debug session-router**.

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.

Command history

SmartWare 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

SmartWare CLI provides a number of Emacs-style command shortcuts that facilitate editing of the command line. Command editing shortcuts are summarized in [table 3](#) on page 44. The syntax **Ctrl-p** means press the **p** key while holding down the keyboard’s “Control” key (sometimes labeled *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.

Table 3. Command edit shortcuts

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.
Ctrl-f or <right-arrow>	Move cursor forward one character.
Ctrl-b or <left-arrow>	Move cursor backward one character.
Esc f	Move cursor forward one word.
Esc b	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.
Esc d	Delete word.
Ctrl-c	Quit editing the current line.
Ctrl-l	Refresh (redraw) the display.
Ctrl-t	Transpose characters.

Table 3. Command edit shortcuts (Continued)

Keyboard	Description
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. E.g. Pression 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 expresion, press Ctrl-v immediately followed by the question mark (?).

Chapter 4 **Accessing the CLI**

Chapter contents

Introduction	48
Accessing the SmartWare CLI task list.....	48
Accessing via the console port	49
Console port procedure	49
Accessing via a Telnet session	50
Telnet Procedure	51
Log onto the SmartWare	51
Selecting a secure password	52
Configure operators and administrators	52
Factory preset administrator account	52
Creating an operator account	52
Creating an administrator account	53
Displaying the CLI version	54
Displaying account information	54
Switching to another account	54
Checking identity and connected users	55
Ending a Telnet or console port session	56

Introduction

SmartNode products are designed for remote management and volume deployment. The management and configuration of SmartNodes is therefore based on IP network connectivity. Once a SmartNode 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 SmartWare commands via the command line interface (CLI), to obtain help, to change operator mode, and to terminate a session. You can access a SmartNode as follows:

- Directly, via the console port (by using a terminal directly connected to a SmartNode)
- Remotely, via the IP network (by using a Telnet application)

The ports available for connection and their labels for each SmartNode model are shown in the getting started guide that came with your SmartNode system.

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 timesaving tool is command completion. If you type part of a command and then press the **<tab>** key, the SmartWare 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 41.



Although SmartWare supports concurrent sessions via Telnet or the console port, we do not recommend working with more than one session to configure a specific SmartNode.

Accessing the SmartWare CLI task list

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

- Accessing via the console port (see [page 49](#))
- Accessing via a Telnet session (see [page 50](#))
- Logging on to the SmartWare (see [page 51](#))
- Selecting a secure password (see [page 52](#))
- Configuring operators and administrators (see [page 52](#))
- Displaying the CLI version (see [page 54](#))
- Displaying account information (see [page 54](#))
- Switching to another log-in account (see [page 54](#))
- Checking identity and connected users (see [page 55](#))
- Ending a Telnet or console port session (see [page 56](#))

Accessing via the console port

To access a SmartNode via its console port, the host computer must be connected directly to the console port (labeled CONSOLE) with a serial cable (see [figure 8](#)). The host must use a terminal emulation application that supports serial interface communication.



Figure 8. Setup for initial configuration via the console port

Note You do not need to configure IP settings if you access the SmartNode 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 that came with your SmartNode system.
2. Configure your serial terminal for 9600 baud, 8 data bits, no parity, 1 start bit, 1 stop bit, and no flow control.
3. Connect the serial terminal to your SmartNode. Use a serial cable according to *Appendix A* of the getting started guide included with your SmartNode device.
4. Power on your SmartNode. 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.
5. Proceed with logging in.

Accessing via a Telnet session

This is the most commonly used method for connecting to a SmartNode. The Telnet host accesses the SmartNode via its network interface. A host can be connected directly to the ETH 1 port (LAN) with a crossover cable (see [figure 9](#), part A) or through an Ethernet hub with two straight cables (see [figure 9](#), part B).

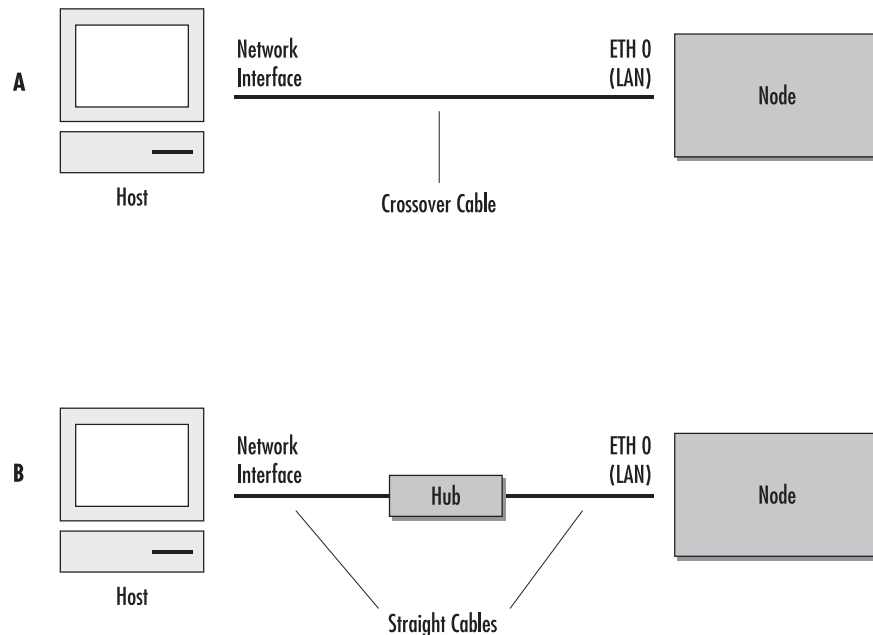


Figure 9. Setup for initial configuration via an Ethernet port

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.

The host must have a valid IP address configured in the same subnet as the SmartNode. [Table 4](#) lists the default IP address and network mask of the Ethernet ports of the SmartNode.

Table 4. Default IP address configuration

Port	IP Address	Network Mask
ETH 0	172.16.40.1 (formerly 10.0.0.10)	255.255.0.0/16
ETH 1	192.168.1.1 (formerly 10.0.0.10)	255.255.255.0/24 (formerly 244.255.0.0/16)

Note The default IP addresses listed in [table 4](#) apply to an operating scenario compatible with the factory configured settings of the SmartNode. If your operating requirements are significantly different, your SmartNode may have different default IP addresses. Check SmartWare release notes for more details.

Telnet Procedure

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

1. Set up the SmartNode as described in the getting started guide included with your SmartNode device.
2. Connect the host (PC) or hub to the ETH 1 (LAN) port of your SmartNode with crossover or straight-thru cables, according to *Appendix A* of the getting started guide included with your SmartNode device.
3. Power on your SmartNode and wait until the *Run* LED lights.
4. Be sure that the IP address and subnet mask of your host are in the same address range as the ETH 1 (LAN) port of your SmartNode.
5. Open a Telnet session to the ETH 1 (LAN) port with the IP address 10.1.0.10 of your SmartNode.
6. Proceed with logging in.

Log onto the SmartWare

Accessing your SmartNode 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 in [figure 10](#). The window header bar shows the IP address of the target SmartNode.

A factory preset administrator account with name *administrator* and an empty password is programmed into the SmartWare at the factory. For that reason, use the name *administrator* after the login prompt and simply press the <Enter> key after the password prompt.

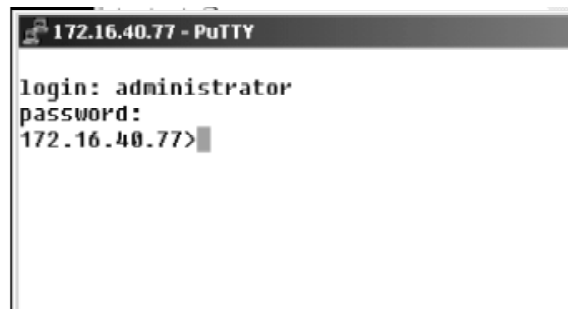


Figure 10. Login display

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 screen in [figure 10](#), such as the IP address in the system prompt and window header bar, may be different on your SmartNode device.



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: *3Bmshtr*

You are probably asking yourself, "How am I going to remember that?" It's easy, the password above is an acronym taken from: "3 blind mice see how they run." Making a good password is that easy! But please, don't use the above example password for your SmartNode device.

Configure operators and administrators

To secure the system, as well as to enable remote access to the system, you must create operator and administrator login accounts. These accounts are valid system-wide. Operators and administrators can log in to the SmartWare via the console or through Telnet.

Note Only administrators are allowed to create new administrator and operator accounts.

Factory preset administrator account

At the beginning of setup, SmartWare contains a factory preset administrator account with the name *administrator* and an empty password. After adding a new administrator account, the factory preset administrator account is automatically deleted and only the newly created administrator account is available. You can create more than one administrator account, but there has to be at least one administrator account defined. If, for some reason, the last administrator account is deleted, SmartWare automatically recreates the factory preset administrator account with the name *administrator* and an empty password.

Creating an operator account

Operators do not have the privileges to run the **enable** command and therefore cannot modify the system configuration. Operators can view partial system information.

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

Mode: Operator execution

Step	Command	Purpose
1	node>enable	Enters administration execution mode
2	node#configure	Enters configuration mode
3	node(cfg)# operator name password password	Creates a new operator account <i>name</i> and password <i>password</i>
4	copy running-config startup-config	Saves the change made to the running configuration of the SmartNode, 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.

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

Creating an administrator account

Administrators can run the **enable** command and access additional information within the SmartWare configuration modes. Therefore administrators can modify the system configuration, as well as view all relevant system information.

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

Mode: Operator execution

Step	Command	Purpose
1	node>enable	Enters administration execution mode
2	node#configure	Enters configuration mode
3	node(cfg)# administrator name password password	Creates a new administrator account <i>name</i> and password <i>password</i>
4	node(cfg)#copy running-config startup-config	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*.

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

Displaying the CLI version

This procedure displays the version of the currently running SmartWare CLI.

Mode: Operator execution

Step	Command	Purpose
1	node>show version cli	Displays the CLI version

Example: Displaying the CLI version

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

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

Displaying account information

You can use the **show** command in the SmartWare 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	node#show accounts	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.

```
SN#show accounts
administrator accounts:
  super
operator accounts:
  support
```

Switching to another account

A user can use the **su** command to switch from one user account to working in another. With this command, a user can change from his current account to another existing account 'name'. After executing **su** with the account name to which the user wants to change as argument, he must enter the password of the particular account to get privileged access.

Mode: Administrator or operator execution

Step	Command	Purpose
1	node>su account-name	Changes to the user account <i>account-name</i> .

Example: Switching to Another Account

The following example shows how to change from your current user account to an administrator account, starting from the operator execution mode. In the example below the **who** command is used to check the identity within both accounts

```
login: support
password: <password>
SN>who
You are operator support
SN>su super
Enter password: <password>
SN>who
You are administrator super
```

Checking identity and connected users

The **who** command displays who is logged in or gives more detailed information about users and process states. 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, state, idle time, 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	node#who	Shows more detailed information about the users ID, name, state, idle time and location
<i>or</i>		
	node>who	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 and process states, depending on the execution mode in which you are working.

Used in administrator execution mode:

```
SN#who
  ID  User name      State  Idle      Location
*   0  administrator  exec    00:00:00  172.16.224.44:1160
    1  support         exec    00:01:56  172.16.224.44:1165
```

Note The “*” character identifies the user executing the **who** command. *ID* represents the ID of the account. *State* represents the actual running condition of the user, which can be logout, login, exec and config.

Used in operator execution mode:

```
SN>who
You are operator support
```

Ending a Telnet 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	node>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.

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

After confirming the dialog with “yes”, the Telnet session to the SmartNode is terminated and the Telnet application window on your host closes.

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

Chapter 5 **Establishing basic IP connectivity**

Chapter contents

Introduction	58
IP context selection and basic interface configuration tasks	58
Entering the IP context, creating IP interfaces and assigning an IP address	58
Defining IP Ethernet encapsulation and binding an IP interface to a physical port	59
Activating a physical port	59
Displaying IP interface information	60
Deleting IP interfaces	61
Examples	62
Setting up an IP interface on an Ethernet port	62

Introduction

This chapter explains how to establish network-based connections to and from your SmartNode using IP interfaces and Ethernet ports. You can configure basic IP connectivity in the *context IP* and the subsidiary *interface* command modes. For a complete description of the IP context and interface configuration related commands referred to in this chapter, see chapter 10, “IP context overview” on page 117, and chapter 11, “IP interface configuration” on page 125.

The chapter includes the following sections:

- IP context selection and basic interface configuration tasks
- Examples (see [page 62](#))

The predefined IP context in SmartWare contains the functionality of a classic IP router. Within the IP context, packets are routed between IP interfaces according to the routing table. The following sections guide you through all the steps necessary to establish network-based IP connectivity to and from your SmartNode.

IP context selection and basic interface configuration tasks

The following are the basic tasks involved in configuring an IP context, the related interfaces, and ports:

- Entering the IP context, creating IP interfaces and assigning an IP address
- Defining IP Ethernet encapsulation and binding an IP interface to a physical port (see [page 59](#))
- Activating the physical port (see [page 59](#))
- Displaying IP interface information (see [page 60](#))
- Deleting IP interfaces (see [page 61](#))

After you have entered the IP context and performed the basic configuration tasks, it is possible to configure additional protocols and services such as RIP, ICMP, and NAT for your IP context.

Entering the IP context, creating IP interfaces and assigning an IP address

SmartWare application software running on your SmartNode has a predefined IP context, which has to be selected for the configuration procedure. An IP interface name can be any arbitrary string of not more than 25 characters. Use self-explanatory names for your IP interfaces which reflect their usage. Each IP interface needs its explicit IP address and an appropriate net mask to be set.

Mode: Configure

Step	Command	Purpose
1	node(cfg)#context ip router	Enters the predefined IP context configuration mode.
2	node(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 <i>name</i> you have just created.
3	node(if-ip)[name]#ipaddress ip-address netmask	Sets the IP address <i>ip-address</i> and <i>netmask</i> for the interface <i>name</i>

Example: Enter IP context, create IP interfaces, and set IP address and netmask

The procedure below assumes that you want to create an IP interface named *lan*, with an IP address of *192.168.1.3* and a net mask of *255.255.255.0*. Use the following commands in configuration mode to select the IP context and create the IP interface.

```
SN(cfg)#context ip router
SN(ctx-ip)[router]#interface lan
SN(if-ip)[lan]#ipaddress 192.168.1.3 255.255.255.0
```

Defining IP Ethernet encapsulation and binding an IP interface to a physical port

Before an IP interface is accessible, you must define the IP Ethernet encapsulation for the related port. It is assumed that you would like to define the IP Ethernet encapsulation for port *port* on slot *slot*. Before an IP interface can be used, it needs to be bound to a physical port of your SmartNode. The SmartNode has one or more expansion slots that can have one or more ports. Specifying a port unambiguously means that you must define the slot in which it is located. It is assumed that you would like to bind the IP interface *name* to port *port* of slot *slot*.

Mode: Configure

Step	Command	Purpose
1	node(cfg)#port ethernet <i>slot port</i>	Enters port configuration mode and selects the Ethernet port <i>port</i> on slot <i>slot</i> , on which use the IP Ethernet encapsulation and to which bind an IP interface.
2	node(prt-eth)[slot/port]#encapsulation ip	Sets IP Ethernet encapsulation for port <i>port</i> on slot <i>slot</i>
3	node(prt-eth)[slot/port]#bind interface <i>name router</i>	Binds the interface name to port <i>port</i> on slot <i>slot</i> to the IP context named <i>router</i> , which is the IP router context

Example: Define IP Ethernet encapsulation and bind IP interface to physical port

It is assumed that you would like to set the IP encapsulation for the Ethernet port *0* on slot *0* and bind the already defined IP interface *lan* to the same physical port. Use the following commands in port Ethernet mode.

```
SN(ctx-ip)[router]#port ethernet 0 0
SN(prt-eth)[0/0]#encapsulation ip
SN(prt-eth)[0/0]#bind interface lan router
```

Activating a physical port

After completing all the settings for the IP interface, you must activate the physical port. The SmartWare default status for any port is disabled. In SmartWare terminology, any port is in the shutdown state unless it is activated by command.

Using the command **show port ethernet *slot port*** lists the actual status for the selected physical port. The following listing shows the port Ethernet information for port 0 on slot 0, which is in the shutdown state as indicated by the current state CLOSED.

```
SN(prt-eth)[0/1]#show port ethernet 0 0

Ethernet Configuration
-----

Port          : ethernet 0 0 0
State         : CLOSED
MAC Address   : 00:30:2B:00:1D:D4
Speed        : 10Mbps
Duplex        : Half
Encapsulation : ip
Binding       : wan@router
Frame Format   : standard
Default Service: 0
```

To activate a port for operation, you must remove the shutdown status of the port. That means you must change the state of the port to OPENED. To activate a physical port, use the **no shutdown** command in port configuration mode.

Step	Command	Purpose
1	node(ctx-ip)[router]#port ethernet <i>slot port</i>	Enters port configuration mode and selects the Ethernet port <i>port</i> on slot <i>slot</i> , which is to be activated
2	node(prt-eth)[<i>slot/port</i>]#no shutdown	Activates the physical port <i>port</i> on slot <i>slot</i> for operation

Example: Activating the physical port

It is assumed that you would like to activate the physical port 0 on slot 0, for which you use the following commands in port configuration mode.

```
SN(cfg)#port ethernet 0 0
SN(prt-eth)[0/0]#no shutdown
```

At this point, your SmartNode has a running IP interface on Ethernet port 0 on slot 0, which uses IP encapsulation.

Displaying IP interface information

You can display information for all the configured IP interfaces by using the **show** command. The command lists relevant information for every IP interface. The IP interfaces are identified by the name.

Mode: Configure

Step	Command	Purpose
1	node(cfg)#show ip interface	Displays IP interface information

Example: List existing IP interfaces

You can display IP interface information by using the **show ip interface** command in configuration mode. In the following example, only the information available for IP interface *lan* is displayed. Depending on the number of defined IP interfaces, the output of the **show ip interface** command can be longer.

```
SN(ctx-ip)[router]#show ip interface
-----
Context:          router
Name:            lan
IP Address:      192.168.1.3 255.255.255.0
P2P:            point-to-point
MTU:            1500
ICMP router-discovery: enabled
ICMP redirect:  send only
State:          OPENED
Binding:        ethernet 0 0 0/ethernet/ip
```

An easy way to list existing interfaces is by using the **interface** command followed by a “?” in the IP context configuration mode, which creates a list of all the defined IP interfaces.

```
SN(cfg)#context ip router
SN(ctx-ip)[router]#interface <?>
<interface>      New interface
external         Existing interface
internal         Existing interface
lan              Existing interface
wan              Existing interface
```

Deleting IP interfaces

It is often necessary to delete an existing interface in the IP context. The procedure described below assumes that you would like to delete the IP interface *name*. Use the **no** argument to the **interface** command as in the following demonstration in IP context configuration mode.

Mode: Context IP

Step	Command	Purpose
1	node(ctx-ip)[router]#no interface name	Deletes the existing IP interfaces <i>name</i>

Example: Delete IP interfaces

The procedure described below assumes that you would like to delete the IP interface named *external*. Use the following commands in IP context mode.

1. List the existing interfaces:

```
SN(ctx-ip)[router]#interface <?>
<interface>      New interface
external         Existing interface
internal         Existing interface
lan              Existing interface
wan              Existing interface
```

2. Delete the interfaces named *external* with the **no interface** command, with the interface name as argument:

```
SN(ctx-ip)[router]#no interface external
```

3. List the interfaces again to check if the IP interface *external* has been deleted:

```
SN(ctx-ip)[router]#interface <?>
<interface>          New interface
internal              Existing interface
lan                   Existing interface
wan                   Existing interface
```

Examples

Setting up an IP interface on an Ethernet port

The following example shows all required configuration steps, which end in an activated IP interface on Ethernet port 0 on slot 0. [Figure 11](#) shows the relation between the IP interface *lan* and the Ethernet port 0 on slot 0. The configuration procedure below starts in the operator execution mode:

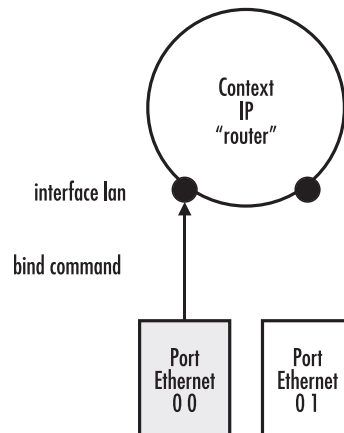


Figure 11. Relation between IP Interface *lan* and Ethernet port 0 on slot 0

1. Select the context IP mode for the required IP interface configuration.

```
SN>enable
SN#configure
SN(cfg)#context ip router
```

2. Create a new interface *lan*, for which both the IP address and net mask are specified.

```
SN(ctx-ip)[router]#interface lan
SN(if-ip)[lan]#ipaddress 192.168.1.3 255.255.255.0
```

3. Select the Ethernet port 0 on slot 0; set the medium to 10 Mbps in half-duplex mode, and choose the IP encapsulation for this port.

```
SN(if-ip)[lan]#port ethernet 0 0
SN(prt-eth)[0/0]#medium 10 half
```



```
SN(prt-eth)[0/0]#encapsulation ip
```

4. Bind the interface *lan* you just defined to the Ethernet port, and then activate the port.

```
SN(prt-eth)[0/0]#bind interface lan router  
SN(prt-eth)[0/0]#no shutdown
```

5. Store the configuration settings in the startup configuration so as to be available after the next system reboot.

```
SN(prt-eth)[0/0]#copy running-config startup-config
```


Chapter 6 **System image handling**

Chapter contents

Introduction	66
Memory regions in SmartWare.....	66
Boot procedure.....	68
Bootloader (for SmartNode 1000 and 2000 Series)	69
Start Bootloader and login	69
Main shell and domains	69
Route Table Manager (RTM)	70
Download Agent	71
Diagnostic	72
Bootloader (for SmartNode 4110/4520 Series).....	73
Start Bootloader	73
Start-up with factory configuration	74
Load a new application image (SmartWare) via TFTP	74
Load a new application image (SmartWare) via the serial link	76
Factory configuration	76
System image handling task list	77
Displaying system image information	77
Copying system images from a network server to Flash memory	78
Copying driver software from a network server to Flash memory	79

Introduction

This chapter describes how to load and maintain system images and driver software. System images contain the application image and driver software images. The application image represents the software running SmartWare, which must be stored in the persistent region of the memory. Driver software images contain software that must also be stored in the persistent region of the memory. Driver software images are used for optional PMC interface cards.

This chapter includes the following sections:

- Memory regions in SmartWare
- Boot procedure and bootloader (see [page 68](#))
- Factory configuration (see [page 76](#))
- System image handling task list (see [page 77](#))

Patton SmartNode devices are shipped with a default system image that is stored in persistent flash memory. The system image contains the application image and driver software images that comprise the SmartWare. In addition, a factory configuration is loaded into the SmartNode at the factory that sets initial SmartWare parameters.

Operational configuration files that you create are stored in SmartNode flash memory. Backup copies can also be stored on a remote server. Transferring configuration files between the flash memory and a remote server requires using the *trivial file transfer protocol* (TFTP). The TFTP server must be accessible through one of the SmartNode IP interfaces. TFTP cannot be used from the console interface.

The following sections focus on SmartWare memory regions, as well as the software components you can copy into the memory or move between a TFTP server and the memory of the SmartNode. Since SmartWare uses a specific vocabulary in naming those software components, refer to appendix A, “[Terms and definitions](#)” on page 517 to ensure that you understand the concepts.

Memory regions in SmartWare

The SmartNode memory SmartWare uses is divided into several regions as shown in [figure 12](#) on page 67. A remote TFTP server is used for uploading or downloading the configurations, application, and driver software images to or from the SmartNode’s memory. In SmartWare command syntax, you must prefix the file path of a file on the TFTP server that is used for image upload or download with *tftp:*, followed by the absolute file path starting from the root directory of the TFTP server.

The flash memory persistently stores data it contains and has two logical regions called *flash:* and *nvrn:*, which are used as follows:

- The application image, a bootloader image and one or more driver software images must be stored in the logical region *flash:* of the flash memory.
- Configuration files must be stored in the logical region *nvrn:* of the flash memory. The factory default configuration is always loaded and may be restored by pressing the SmartNode reset button; see the getting started guide that came with your SmartNode. The startup, or user-specific configuration, is also stored in *nvrn:*.

The factory configuration is read-only. It is contained in the logical region *nvrn:* of the SmartNode. It can be used if no user-specific configuration is available to start-up SmartWare with a minimal functionality. This configuration is named *factory-config* in SmartWare terminology.

A dedicated user-specific configuration must be created and stored in the flash memory. This configuration defines the user's desired system functionality and is used to start-up the system under normal conditions. This configuration must be stored as *default-config* in the logical region *nvram:* of the flash memory. Any configuration stored in the logical region *nvram:* can be copied to a remote server by using TFTP.

Since configurations cannot be executed from the persistent memory, any configuration to be used must be copied into the volatile memory of the SmartNode prior to operation. This procedure takes place after the system bootstrap, where the application image (i.e. SmartWare) is started and a configuration must be available. Shortly before SmartWare is fully started up, the configuration *startup-config* is copied from the logical region *nvram:* of the flash memory as the *running-config* into the volatile memory *system:* of the SmartNode. The volatile memory *system:* is a logical region in the random access memory (RAM) of the SmartNode.

Changing any settings during operation of a SmartNode alters the running configuration, i.e. the one named *running-config* in the volatile memory *system:*. In order to have such modifications available after the next system start, the running configuration must be stored back as *startup-config* into the persistent memory *nvram:*. Furthermore, it is possible to backup the running-config located in the volatile memory *system:* into the persistent memory *nvram:* or on a remote TFTP server, by using a user-defined name.

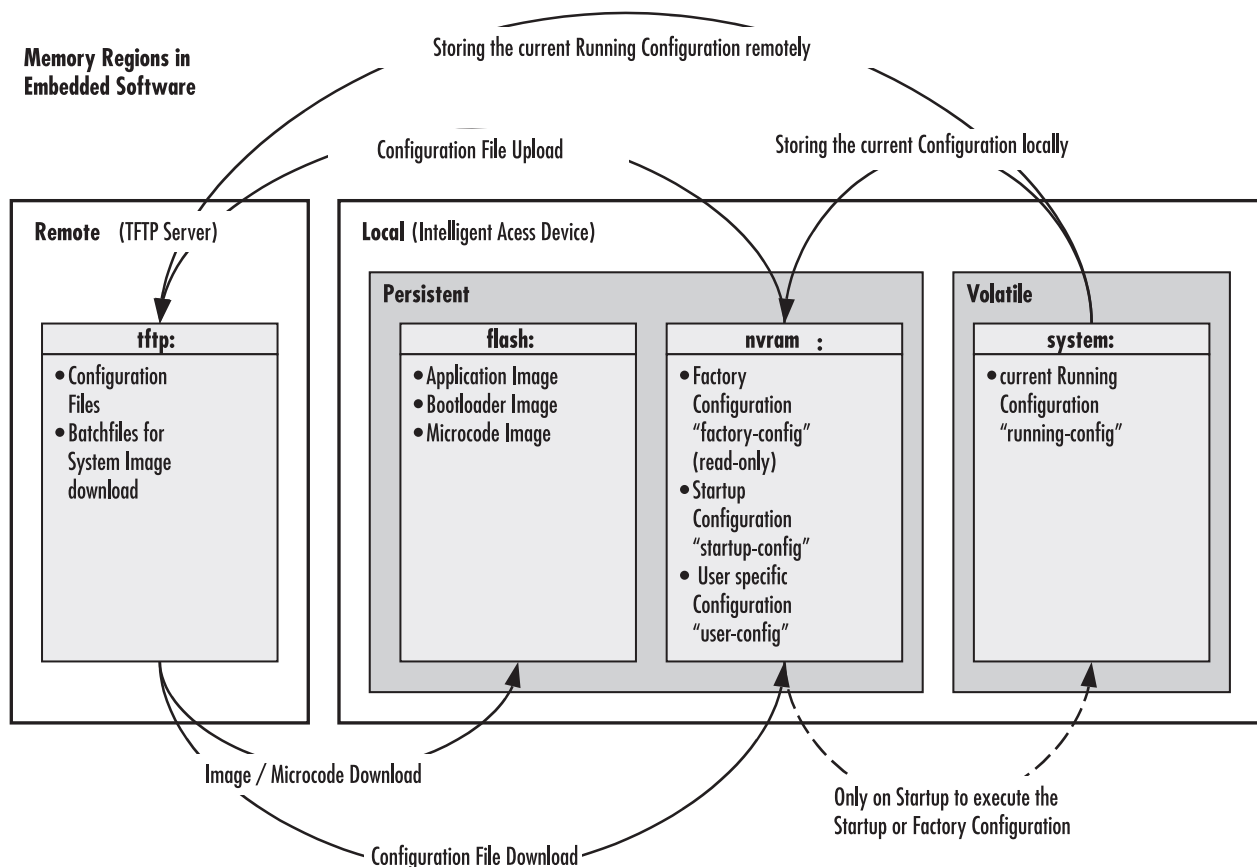


Figure 12. SmartNode memory regions logically defined in SmartWare

Boot procedure

During a normal boot procedure of a SmartNode, the bootstrap application checks for an application image in the persistent memory of the logical region *nvram*:. The application image is then executed, i.e. the SmartWare is started module by module. Shortly before SmartWare is fully started up, the configuration *startup-config* is copied from the logical region *nvram*: of the flash memory as *running-config* into the volatile memory *system*: and it is used to parameterize SmartWare. Figure 13 illustrates the boot procedure.

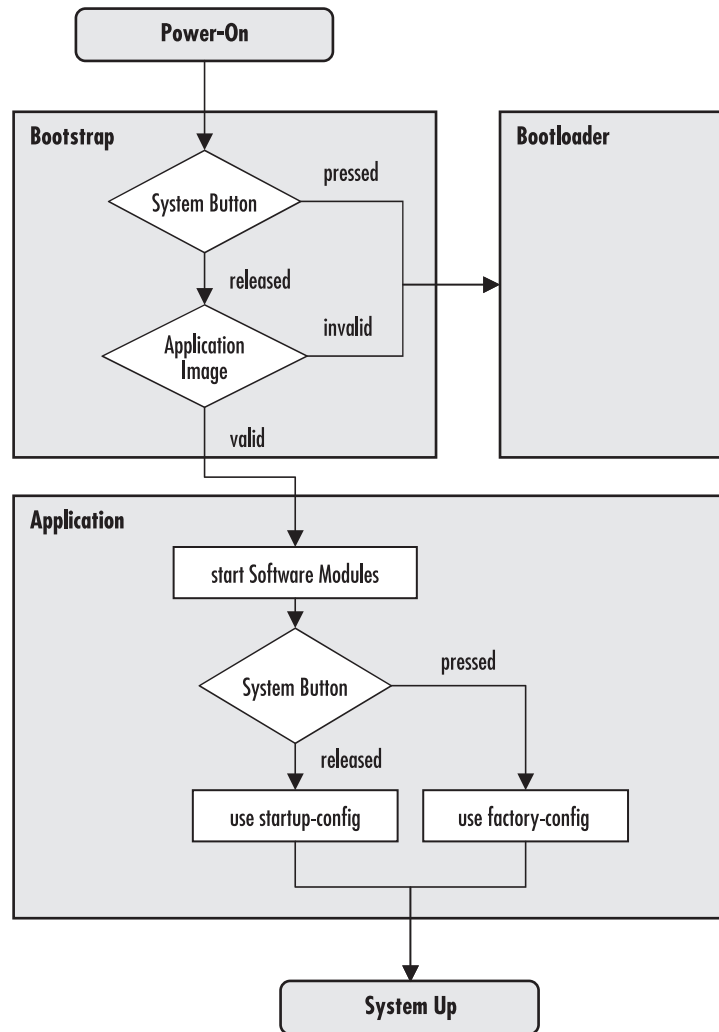


Figure 13. Boot procedure

There are two situations during bootstrap when the bootloader takes control:

- The bootstrap application checks the status of the Reset button (not available for SN4xxx) on the back panel of the SmartNode; if the user has pressed the system button, it launches the bootloader
- If a valid application image is not available

The bootloader ensures that basic operations, network access, and downloads are possible in case of interrupted or corrupted application image downloads. After downloading an application image, the bootstrap only

switches to the newly loaded application image if it is valid. Otherwise, the bootstrap still uses the previous application image.

If the application image is valid, it is started and SmartWare is brought into operation module by module. During this system initialization phase (when the message *Press reset button to restore factory defaults...* appears on the console screen), the status of the reset button on the back panel of the SmartNode is checked. If the button has been pressed, the factory configuration is loaded into the volatile memory and is used to parameterize the SmartWare (not available for SN4xxx). If the button has not been pressed, the startup configuration is loaded into the volatile memory and is used to parameterize the SmartWare.

Bootloader (for SmartNode 1000 and 2000 Series)

Start Bootloader and login

To start the Bootloader explicitly, power on the SmartNode *while* pressing the reset button. All front LEDs will light up. Keep pressing the reset button until the BRI/Ethernet LEDs on SmartNode 1x00 and the ACT LED on SmartNode 2x00 are off. When the bootloader starts, the BRI LEDs on SmartNode 1x00 and the ACT LED on SmartNode 2x00 are blinking. Open a Telnet connection to the SmartNode via either one of Ethernet interfaces and the Login display shown in [figure 14](#) will appear. Use the credentials *admin/patton* to login.

Note The Bootloader does *not* support the console interface. The Ethernet interfaces preserve the IP addresses, IP masks, and default gateway that they had before starting the bootloader.

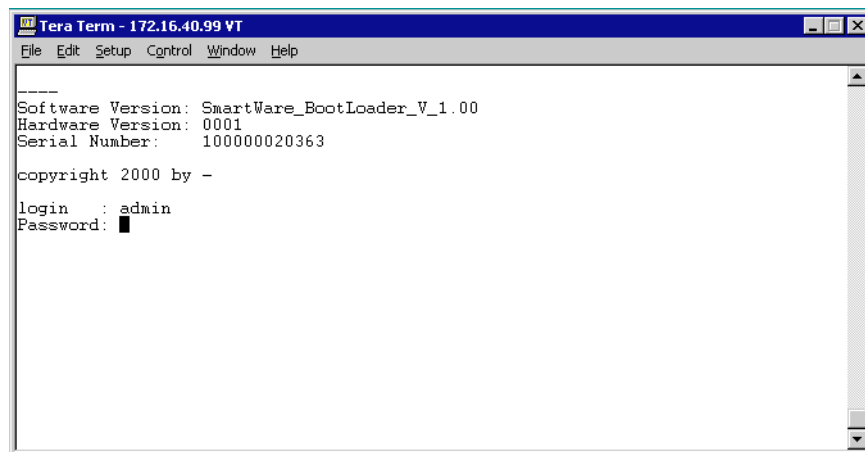


Figure 14. Login display

Main shell and domains

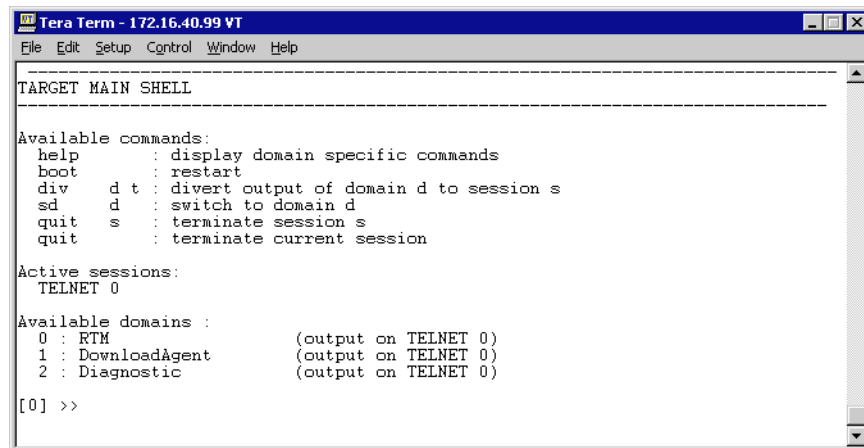
After login, you access the Main Shell (see [figure 14](#)). You can select from three available domains:

- Route Table Manager (RTM)
- Download Agent
- Diagnostic

The available command set is shown in [table 5](#).

Table 5. Main shell command set

Command	Function
?	Displays the main menu with available commands, domains and active sessions.
help	Displays a list of the commands available in the current domain.
boot	Restarts the system.
div d s	Diverts the output of domain d to another session s, i.e. to Telnet interfaces.
sd d	Switches to another domain. You can display the available domains in the main menu by entering "?". The "d = 0" command invokes the Route Table Manager; "d = 1" invokes the Download Agent; "d = 2" invokes the Diagnostics.
quit/quit s	Terminates the current session/the session at Telnet interfaces.



```

Tera Term - 172.16.40.99 VT
File Edit Setup Control Window Help
-----
TARGET MAIN SHELL
-----
Available commands:
help      : display domain specific commands
boot      : restart
div d t   : divert output of domain d to session s
sd d      : switch to domain d
quit s    : terminate session s
quit      : terminate current session

Active sessions:
TELNET 0

Available domains :
0 : RTM                (output on TELNET 0)
1 : DownloadAgent      (output on TELNET 0)
2 : Diagnostic          (output on TELNET 0)

[0] >>

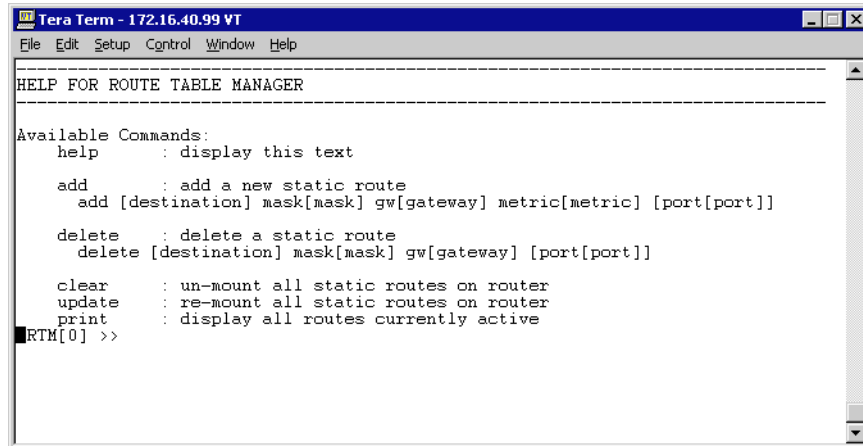
```

Figure 15. Main shell

Route Table Manager (RTM)

To access the Route Table Manager, type `sd 0`.

Type **help** to display a list of commands that are available in the RTM domain (see figure 16).



```
Tera Term - 172.16.40.99 VT
File Edit Setup Control Window Help
-----
HELP FOR ROUTE TABLE MANAGER
-----
Available Commands:
  help      : display this text

  add       : add a new static route
             add [destination] mask[mask] gw[gateway] metric[metric] [port[port]]

  delete    : delete a static route
             delete [destination] mask[mask] gw[gateway] [port[port]]

  clear     : un-mount all static routes on router
  update   : re-mount all static routes on router
  print    : display all routes currently active
RTM[0] >>
```

Figure 16. Route Table Manager display

To add a new static route, use the command **add**.

For example:

```
add 1.2.3.4 mask 255.255.0.0 gw 1.2.3.10 metric 0
```

You can delete a route by using the command **delete**.

For example:

```
delete 1.2.3.4 mask 255.255.0.0 gw 1.2.3.10
```

To deactivate/activate all static routes, use the commands **clear** followed by **update**.

You can display a list of all routes currently active by typing **print**.

Download Agent

To access the Download Agent, type **sd 1**.

Type **help** to display a list of commands that are available in the Download Agent domain (see [figure 17](#)).

```

Tera Term - 172.16.40.99 VT
File Edit Setup Control Window Help
-----
HELP FOR DOWNLOAD AGENT
-----
ssip <Server IP address>:
  sets TFTP server address (e.g. ssip 172.16.1.175)
gsip:
  displays TFTP server address
strc <number of retries>:
  sets the TFTP retry count
gtrc:
  gets the TFTP retry count
stcf <true|false>:
  sets the TFTP continue flag
gtcf:
  gets the TFTP continue flag
sdfll <time in minutes>:
  sets the download file lifetime
gdfll:
  gets the download file lifetime
download <server base path> <file name>:
  downloads and reads the config file, then downloads what is listed
  in the config file. (e.g. download /home/config/in1200 my_config)
DownloadAgent[0] >>

```

Figure 17. Download Agent display

The Download Agent allows you to set and read different TFTP server settings:

- IP address (**ssip**, **gsip**)
- Retry count (**strc**, **gtrc**)
- Continue flag (**stcf**, **gtcf**)
- Download file lifetime (**sdfll**, **gdfll**)

You can use the command **download** to download an application image or a configuration file from the TFTP server, for example:

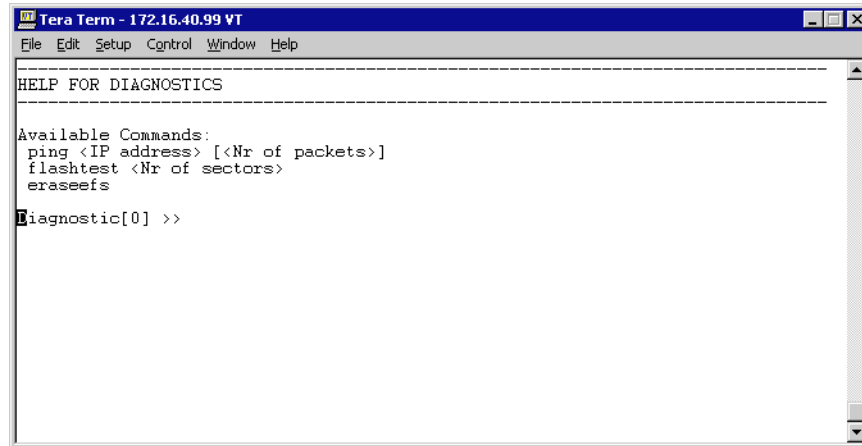
```
download /SmartWare/Sn1xxx/Vx/R2.10/BUILD21215 b
```

where */SmartWare/...* is the path to the directory where the application image (Build) is stored, relative to the configured TFTP root, and 'b' is the batch file that tells the Download Agent which files to download.

Diagnostic

To access the Diagnostic domain, type **sd 2**.

Type **help** to display a list of the available commands in the Diagnostic domain (see [figure 18](#)).

The image shows a screenshot of a Tera Term terminal window. The title bar reads "Tera Term - 172.16.40.99 VT". The menu bar includes "File", "Edit", "Setup", "Control", "Window", and "Help". The main text area displays "HELP FOR DIAGNOSTICS" followed by a list of available commands: "ping <IP address> [<Nr of packets>]", "flashtest <Nr of sectors>", and "eraseefs". The prompt "Diagnostic[0] >>" is visible at the bottom of the text area.

```
Tera Term - 172.16.40.99 VT
File Edit Setup Control Window Help
-----
HELP FOR DIAGNOSTICS
-----
Available Commands:
ping <IP address> [<Nr of packets>]
flashtest <Nr of sectors>
eraseefs
Diagnostic[0] >>
```

Figure 18. Diagnostic Display

The command **ping** allows you to verify the IP connectivity within a network.

You can test the various sectors of the SmartNode flash memory with the command **flashtest**.

Use the command **eraseefs** to delete the contents of the EFS.

Bootloader (for SmartNode 4110/4520 Series)

The SmartNode 4110/4520 Series comes with a new Bootloader, the *RedBoot Bootloader*. It offers new features such as console access to the Bootloader and the capability for downloading application images (e.g. SmartWare) via the serial link of the console.

Start Bootloader

To start the Bootloader, reload the system and press Ctrl-C (when the message *Press ^C to abort boot script, ...* appears on the console screen). The follow prompt will be displayed:

```
RedBoot>
```

Type **help** to display an overview of the available commands.

Note If the cursor keys (up, down, left, right) are not working, use Ctrl-N (for up) and Ctrl-P (for down) instead. Commands can be abbreviated as long as they do not become ambiguous.

Start-up with factory configuration

Step	Command	Purpose
1	RedBoot> fis load	Copies the SmartWare application image from the persistent memory (flash:) to the volatile memory (RAM) from where it will be executed.
2	RedBoot> go -s factory-config	Starts the SmartWare application telling it to use 'factory-config' as startup configuration. You can also start-up with any other configuration available in the persistent memory (nvram:) by providing its name instead of 'factory-config'.

Load a new application image (SmartWare) via TFTP

The following procedure downloads the application image (SmartWare) for the mainboard. See the note below on how to download the respective CLI description file.

Step	Command	Purpose
1 optional	RedBoot> ip_address -l <i>local_ip_address</i> [/mask_len]	Sets the IP address and subnet mask of the Ethernet interface 0/0 which shall be used to receive the new application image. <i>mask_len</i> is the length of the network address (or the number of 1's within the subnet mask). See Note below.
2 optional	RedBoot> ip_address -g <i>gateway</i>	Sets the IP address of the default gateway.
3 optional	RedBoot> ping -h <i>tftp-server_ip_address</i>	Tests the connectivity to the TFTP server.
4	RedBoot> load -r -v -h <i>host -b</i> <i>base_address file_name</i>	Downloads an application image into the volatile memory (RAM) from where the SmartNode could directly execute it. <i>host</i> : IP address of the TFTP server <i>base_address</i> : memory location where to store the application image. Use the default address 0x1800100 <i>file_name</i> : path and name of the file on the TFTP server. Note: use the image file that contains the whole application, not the image parts.
5	RedBoot> fis delete -n 1	Deletes the first application image. Reply with 'y' to the confirmation request.
6	RedBoot> fis create	Stores the downloaded application image to the permanent memory (flash:). Reply with 'y' to the confirmation request.
7	RedBoot> fis list -l	Checks whether the image has been successfully stored, whether it is the desired Release and Build, and whether it is valid.

Step	Command	Purpose
8	RedBoot> go	Starts the application image that was downloaded into the volatile memory (RAM).

Note With the Bootloader, only the Ethernet interface 0/0 is available. The Bootloader applies the IP address, subnet mask, and default gateway that were last configured by the Bootloader itself or by another application (e.g. SmartWare). If an application configured the Ethernet interface 0/0 to use DHCP, the Bootloader will also use DHCP to learn the interface configuration. It can receive and apply the IP address, subnet mask, default gateway, and default (TFTP) server (transmitted as basic DHCP information 'Next server IP address').

Note This procedure does not download the respective CLI description file. Download it after starting up SmartWare with the following command:
 copy tftp://<tftp_server_address>/<server path>/bl flash:

Example: Downloading and storing a new application image (SmartWare)

```
RedBoot> ip -l 172.16.40.98/19
RedBoot> ip -g 172.16.32.1
RedBoot> ping -h 172.16.32.100
Network PING - from 172.16.40.98 to 172.16.32.100
.....PING - received 10 of 10 expected

RedBoot> load -r -v -h 172.16.32.100 -b 0x1800100 /Sn4xxx/image.bin
Using default protocol (TFTP)
-
Raw file loaded 0x01800100-0x0199ca6b, 1689964 bytes, assumed entry at 0x01800100

RedBoot> fis delete -n 1
Delete image 1 - continue (y/n)? y
... Erase from 0x60030000-0x601cc974: .....
```

```
RedBoot> fis create
Use address 0x01800100, size 1684402 ? - continue (y/n)? y
... Erase from 0x60030000-0x601cb3ba: .....
```

```
... Program from 0x00011eec-0x00011ef4 at 0x60030000: .
... Program from 0x01800100-0x0199b4b2 at 0x60030008: .....
... Program from 0x00011eec-0x00011ef4 at 0x60030000: .
Image successfully written to flash
```

```
RedBoot> fis list -l
Id Address      Length  State      Description
  Entry      Load Addr      Version
-----
1  0x60030000  1693438  valid      SmartWare R2.10 BUILD28015
   0x01800100  0x01800100  V2.10
```

```
RedBoot> go
Starting 'SmartWare R2.10 BUILD28015' at 0x01800100 via 0x01800100
```

Load a new application image (SmartWare) via the serial link

The Bootloader supports the 'X-Modem' and 'Y-Modem' protocols to download application images via the serial link of the console. Do the following to initiate the download:

Step	Command	Purpose
1	RedBoot> load -r -v -m { xmodem ymodem } -b base_address	Downloads an application image into the volatile memory (RAM) from where the SmartNode could directly execute it. 'xmodem' or 'ymodem': Specify the protocol to be used, X-Modem or Y-Modem <i>base_address</i> : memory location where to store the application image. Use the default address 0x1800100 Execute the above RedBoot command first, then start the transfer from the terminal program with the command 'Send file via X-Modem' (or similar).
5	RedBoot> fis delete -n 1	Deletes the first application image. Reply with 'y' to the confirmation request.
6	RedBoot> fis create	Stores the downloaded application image to the permanent memory (flash:). Reply with 'y' to the confirmation request.
7	RedBoot> fis list -l	Checks whether the image has been successfully stored, whether it is the desired Release and Build, and whether it is valid.
8	RedBoot> go	Starts the application image that was downloaded to the volatile memory (RAM).

Note This type of download takes about **25 minutes** (for the SmartNode 1000 series) since it uses a serial link at only 9600 baud.

Factory configuration

SmartNodes are delivered with a *factory configuration* stored in the logical region *nvr* of the memory. It is used to initially parameterize the network and component settings of SmartWare, which makes sense at the very beginning. Moreover, in case of SmartWare malfunction, you can reset to the initial state by reloading the factory configuration. The factory configuration consists of the default settings for the IP networking subsystem.

Once the user-specific configuration is created and stored as startup configuration, the factory configuration is no longer used but it remains in the persistent memory. It is possible to switch back to the factory configuration at any time during the operation of a SmartNode. See section "Boot procedure" on page 68 and section "Start-up with factory configuration" on page 74 for information on how to restore the factory configuration.



Avoid downloading any system image if you do not completely understand what you have to do!

System image handling task list

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

- Displaying system image information
- Copying system images from a network server to the Flash memory (see [page 78](#))
- Copying the driver software from a network server to the Flash memory (see [page 79](#))

Displaying system image information

This procedure displays information about system images and driver software

Mode: Administrator execution

Step	Command	Purpose
1	show version	Lists the system software release version, information about optional interface cards mounted in slots, and other information.

Example: Display system image information

The following example shows the information that is available for a SmartNode 2000 series device with an optional IC-4BRV interface card mounted in slot 2.

```
SN#show version

Product name      : SN2300
Software Version  : SmartWare R3.10 R2.00 BUILD22031
Supplier         :
Provider        :
Subscriber      :

Information for Slot 0:
2500 (Admin State: Application Started, Real State: Application Started)
Hardware Version : 1, 1
Serial number    : 100000021579
PLD Version     : 23010204h
Software Version : SmartWare R2.00 BUILD22031

Information for Slot 1:
this Slot is empty

Information for Slot 2:
IC-4BRV (Admin State: Application Started, Real State: Kernel Started)
Hardware Version : 1
PLD Version     : 170001h
Software Version : Build 24026, min required : Build 24027
Loader Version  : Build 39, min required: Build 39

Information for Slot 3:
this Slot is empty
```

Copying system images from a network server to Flash memory

As mentioned previously, the system image file contains the application software that runs SmartWare; it is loaded into the flash memory at the Patton Electronics Co. factory. Since most of the voice and data features of the SmartNode 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 SmartNode 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 a new SmartWare 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 SmartNode flash memory. To store the system image file, you must use a special download script file. This script file defines how to handle the system image file and where to store it. You cannot download any system image file without an appropriate script file.

Each line in the script file is a command for the CLI of your SmartNode. To download a system image file, which will replace the currently running SmartWare application software, a script file with only one command is necessary.

Comment lines must have a hash character # in column one and can appear anywhere in the script file. Comment lines contain information for administrators or operators who maintain or use the script file.

The following example shows a script file used to download a system image and command line syntax definition file from a TFTP server.

```
# script file for system image download
# Patton Electronics Co. 2001-10-24
image.bin 1369474 21; ver 2300.1,2300.2;
cli.xml
+/flash/cli/spec.xml
*U D
```

Note The script file includes a 32-bit CRC on the last line, displayed as four characters when seen in an ordinary text editor. *Do not delete* the line containing the CRC entry or the download will fail!

You can download the script file with the **copy** command. The **copy** command source defines the TFTP path to the script file and the target is set to use the script parser. After downloading the script file, the system image file and command line syntax definition file download starts automatically.

Mode: Administrator execution

Step	Command	Purpose
1	node(cfg)# copy tftp://node-ip-address/b flash:	Downloads the script file <i>b</i> from the TFTP server at address <i>node-ip-address</i> and starts the system image download process. This progress is visualized with a counter, counting up from 0 to 100% according to the downloaded amount of the file size for each file that needs to be downloaded.

Example: Copy system images from a network server to the Flash memory

The following example shows how to download the driver software image file from the TFTP server at IP address 172.16.36.80. The download is defined by a script file, which has to be downloaded first. After downloading the script file, the driver software image file is downloaded automatically.

```
SN>enable
SN#configure
SN(cfg)#copy tftp://172.16.36.80/sn2300/build22032/b flash:
Completed image download
Completed file download /flash/cli/spec.xml

SN(cfg)#
```

Note When encountering problems due to memory exhaustion (message *Parsing batch file...% APP - OUT OF MEMORY*), shutdown the H.323 gateway prior to initiating the download command as follows (which will temporarily free the required memory): `node(gw-h323)[h323]#shutdown`

After the successful download, either issue the **reload** command (in order to start the IPNode with the new software) or restart the H.323 gateway, thus enabling calls again (with the current software):

```
node(gw-h323)[h323]#no shutdown
```

Copying driver software from a network server to Flash memory

Driver software images contain the driver software to be downloaded into hardware devices such as optional interface cards.

Downloading a driver software image file means storing it permanently at a defined location within the flash memory on the motherboard or in the non-volatile memory of an optional interface card. To download the driver software image file, you must use a special download script file.

The following example shows a script file used to download a driver software image file from a TFTP server for an IC-4BRV interface card.

```
# script file for driver software image download
# Patton Electronics Co. 2001-10-24
;
/IC-4BRVoIP_Vx_R2.00_BUILD24028
+/flash/bin/pmc000216a6
4_ -
```

This script file defines how to handle the driver software image file and where to store it.

Note You cannot download any driver software image file without an appropriate script file.

Mode: Administrator execution

Step	Command	Purpose
1	<code>node(cfg)# copy tftp://node-ip-address/b flash:</code>	Downloads the script file <i>b</i> from the TFTP server at address <i>node-ip-address</i> and starts the driver software image download process. This progress is visualized with a counter, counting up from 0 to 100% according to the downloaded amount of the file size for each file that needs to be downloaded.

Example: Copy driver software from a network server to the Flash memory

The following example shows how to download the driver software image file from the TFTP server at IP address 172.16.36.80. The download is defined by a script file, which has to be downloaded first. After downloading the script file, the driver software image file is downloaded automatically.

```
SN>enable
SN#configure
SN(cfg)#copy tftp://172.16.36.80/ic-4brvoip/build24028/b flash:
Completed file download /flash/bin/pmc000216a6

SN(cfg)#
```

Chapter 7 **Configuration file handling**

Chapter contents

Introduction	82
Understanding configuration files	82
Factory configuration	84
Configuration file handling task list.....	85
Copying configurations within the local memory	86
Replacing the startup configuration with a configuration from Flash memory	87
Copying configurations to and from a remote storage location	89
Replacing the startup configuration with a configuration downloaded from TFTP server	90
Displaying configuration file information	91
Modifying the running configuration at the CLI	91
Modifying the running configuration offline	92
Deleting a specified configuration	93

Introduction

This chapter describes how to upload and download configuration files from and to a SmartNode 1000, 2000, or 4000 Series devices. A configuration file is a batch file of SmartWare commands used in the software modules that perform specific functions of the SmartNode. This chapter also describes some aspects of configuration file management. Refer to chapter 6, “[System image handling](#)” on page 65 for more information.

This chapter includes the following sections:

- Factory configuration (see [page 84](#))
- Configuration file handling task list (see [page 85](#))

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

A configuration file is like a script file containing SmartWare 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 factory 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 SmartNode.
- You can also create a new configuration file or modify an existing one offline. You can copy configuration files from the SmartNode 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 SmartNode network interfaces.

See chapter 4, “[Accessing the CLI](#)” on page 47 for information concerning access to the CLI.

The following sections focus on SmartWare memory regions and software components that can be copied within the memory or uploaded/downloaded between a TFTP server and the memory of the SmartNode. Since SmartWare uses a specific vocabulary in naming those software components, refer to appendix A, “[Terms and definitions](#)” on page 517 to ensure that you understand the concepts. Refer to chapter 6, “[System image handling](#)” on page 65 for a brief description of how SmartWare uses system memory.

Understanding configuration files

Configuration files contain SmartWare commands that are used to customize the functionality of your SmartNode device. During system startup, the SmartWare 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 SmartWare CLI to enter a command during operation of a SmartNode, you alter the running configuration accordingly. In other words, you are modifying a live, in-service system configuration.

Figure 19, shows the characteristics of a configuration file. It is stored on a TFTP server in the file *SN2300_001.cfg* for later download to the SmartNode *SN*. The command syntax used to enter commands with the CLI and add commands in configuration files is identical. For better comprehension, you can add comments in configuration files. To add a line with a comment to your configuration file, simply begin the line with the hash (#) character. The command parser skips everything after the hash character to the end of the line.

```
#-----#
# SmartNode IP and Voice configuration                               #
#-----#
#
# Node:          SN                                               #
# Config:        SN2300_001.cfg                                    #
# Model:         SN2300 0001-0001                                 #
# Serial No.:    100000021579                                     #
# Administrator: LB                                              #
# Date:         12/10/2001                                       #
#
#-----#

# SNTP configuration used for time synchronization
cli version 3.00
 sntp-client
 sntp-client server primary 172.16.1.10 port 123 version 4
 sntp-client poll-interval 600
 sntp-client gmt-offset + 01:00:00

# system definitions
system
 clock-source 1 2
 hostname SN

# IP context configuration
context ip router
 route 0.0.0.0 0.0.0.0 172.19.32.2 1
 route 172.19.41.0 255.255.255.0 172.19.33.250
 route 172.19.49.0 255.255.255.0 172.19.33.250

# interface LAN used for connection to internal network
interface lan
 ipaddress 172.19.33.30 255.255.255.0
 mtu 1500

# interface WAN used for connection to access network
interface wan
 ipaddress 172.19.32.30 255.255.255.0
 mtu 1500

# CS context configuration
context cs switch
 no shutdown

# routing table configuration
routing-table called-e164 rtab
 route 2.. dest-interface telecom-operator
```

```
# interface used to access the PSTN telecom operator
interface isdn telecom-operator
    route call dest-interface h323

# interface used to access the VoIP telecom provider
interface h323 voip-provider
    route call dest-table rtab
    remoteip 172.19.33.60
    bind gateway h323

# H.323 gateway primarily used
gateway h323
    faststart
    no ras
    gatekeeper-discovery auto
    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 19. 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. SmartWare predefines some names for configuration files. These are the factory configuration (*factory-config*), startup configuration (*startup-config*), and running configuration (*running-config*) file names. Refer to appendix A, “[Terms and definitions](#)” on page 517 to learn more about configuration file types.

Factory configuration

Patton SmartNodes are delivered with a *factory configuration* in the logical region *nvr*am. This factory configuration initially parameterizes the most useful network and component settings of SmartWare. Moreover, in case of SmartWare malfunction, resetting to the initial state means possibly reloading the factory configuration. The factory configuration consists of:

- Default settings for the IP networking subsystem
- Default settings for H.323 and SIP gateway subsystem
- Default settings for the quality of service subsystem

Once a user-specific configuration is created and stored as the startup configuration, the factory configuration is no longer used, but still remains in the persistent memory. It is possible to switch back to the factory config-

uration at any time during the operation of a SmartNode configuration. The getting started guide included with your SmartNode device describes the restoration procedure for restoring the default settings.



Avoid downloading any configuration file if you do not completely understand what you have to do! If a configuration file download fails or succeeds only partially your SmartNode device cannot start up without a support intervention at the factory.

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 SmartNode 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 91 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 86](#))
- Replacing the startup configuration with a configuration from the Flash memory (see [page 87](#))
- Copying configurations to and from a remote storing location (see [page 89](#))
- Replacing the startup configuration with a configuration downloaded from the TFTP server (see [page 90](#))
- Displaying configuration file information (see [page 91](#))
- Modifying the running configuration at the CLI (see [page 91](#))
- Modifying the running configuration offline (see [page 92](#))
- Deleting a specified configuration (see [page 93](#))

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 SmartWare as shown in [figure 20](#).

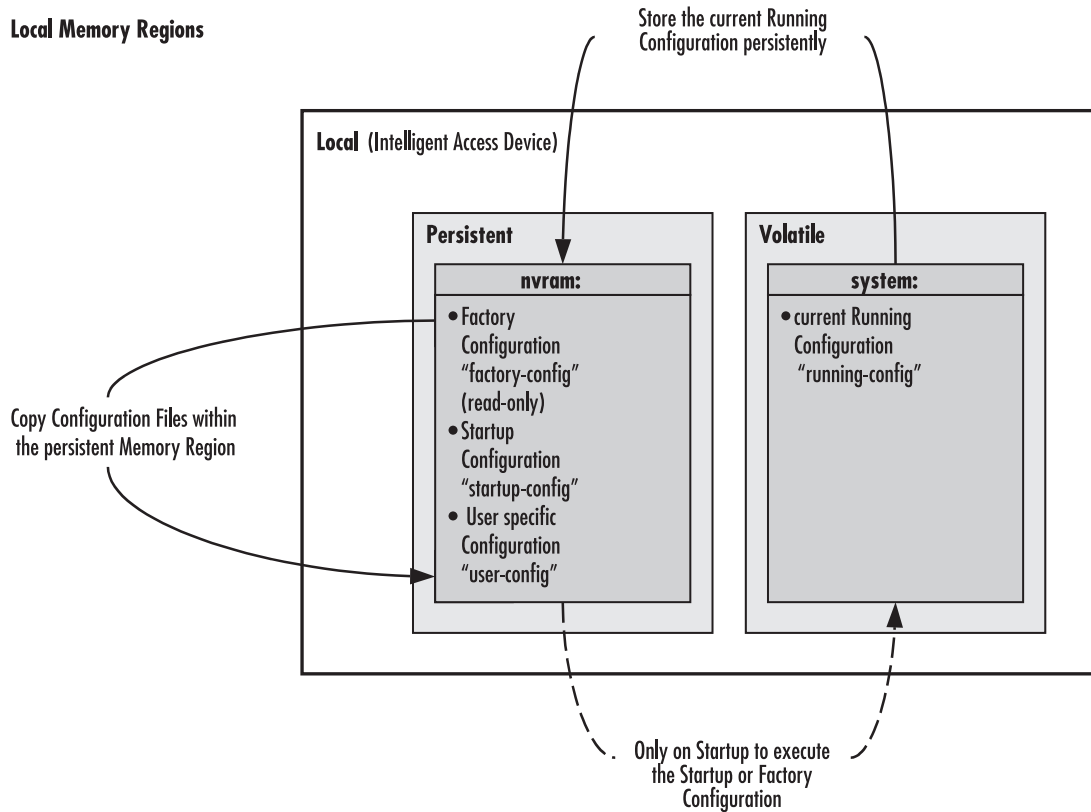


Figure 20. Local memory regions in SmartWare

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 *nvram:*. 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 *nvram:* 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 *nvram:* by using a user-specified name, for conservation or later activation.

As shown in [figure 20](#) the local memory regions are identified by their unique names, like *nvram:*, 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 *nvram:*.

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 three predefined configuration file names for which it is optional to specify the memory region, namely *factory-config*, *startup-config* and *running-config*.

Mode: Administrator execution

Step	Command	Purpose
1	node#copy {factory-config startup-config running-config nvram: source-name } nvram: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 nvram:.

```
SN#copy startup-config nvram: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	node# copy nvram:new-startup startup-config	Replaces the existing persistent startup configuration with the startup configuration <i>new-startup</i> already present in flash memory.

Note It is assumed that the configuration *new-startup* that is present in flash memory was previously copied to the flash memory, e.g. from a TFTP server by using the **copy** command.

Example: Replacing the startup configuration with a configuration from Flash memory

The following example shows how to replace the persistent startup configuration in the flash memory of a SmartNode by overwriting it with the configuration in the file *new-startup* stored in flash memory.

1. Replace the current startup configuration, by using the **copy** command, into the flash memory area where the startup configuration is stored.

```
SN#copy nvram:new-startup startup-config
```

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

```
SN#show startup-config
Startup configuration:
#-----#
# SmartWare R3.10 BUILD24128 #
# 2001-10-25T09:20:42 #
# Generated configuration file #
#-----#
```

```
cli version 3.00
snmp community public rw
```

```
framerelay
exit
```

```
SN#
```

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. Remember the different store locations; they are the local memory in your SmartNode and the remote data store on a server system (see figure 21). A remote storage location is mostly used to store ready configurations for later download to a certain SmartNode. A TFTP server has to be used as a remote data store. From within SmartWare, this 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.

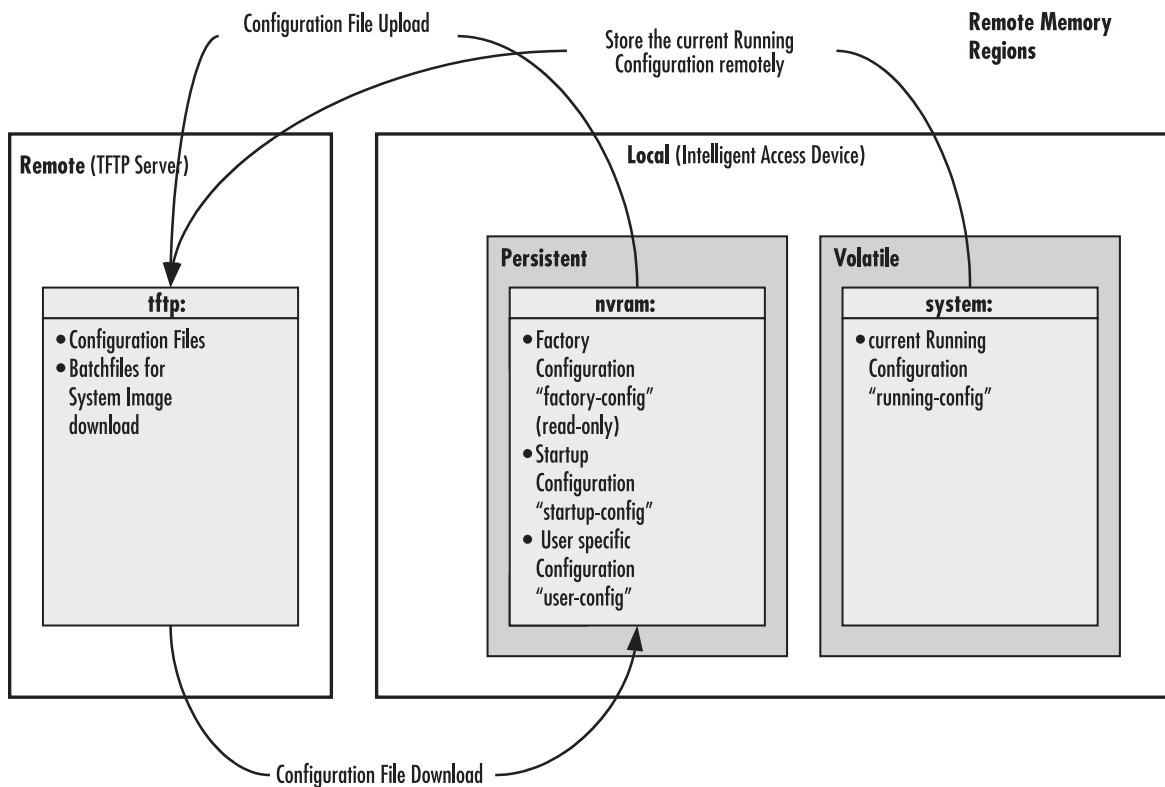


Figure 21. Remote memory regions for SmartWare

Finally, configuration files, i.e. the startup configuration or a user-specific configuration that is stored in the persistent memory region *nvram:* are often uploaded to the remote data store for backup, edit or cloning purposes. The latter procedure is very helpful when you have several SmartNode 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 SmartNode 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 *nvram:* of the target device. After this, the startup configuration is

transferred to the TFTP server, where it can be distributed to other SmartNode 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	node(cfg)# copy tftp://ip-address[:port]/new-startup nvram: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 SmartNode 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.

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

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

```

SN#show nvram:startup-config
Startup configuration:
#-----#
# SmartWare R3.10 BUILD22128 #
# 2001-10-25T09:20:42 #
# Generated configuration file #
#-----#

cli version 3.00
snmp community public rw

framerelay
exit

SN#

```

Displaying configuration file information

This procedure describes how to display information about configuration files

Mode: Administrator execution

Command	Purpose
show nvram:	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

Modifying the running configuration at the CLI

SmartWare 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 SmartNode's operation. When you configure SmartWare by using the CLI, the shell executes the commands as you enter them.

When you log in to a SmartNode by using the CLI, all commands you enter directly modify the running configuration located in the volatile memory region *system:* (or RAM) of your SmartNode. 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 *nvram:* 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 SmartNode by using the **reload** command to cause the system to initialize with the new configuration.

Mode: Administrator execution

Step	Command	Purpose
1	node#configure	Enters administrator configuration mode
2	Enter all necessary configuration commands.	
3	node(cfg)#copy running-config startup-config	Saves the running configuration file as the upcoming startup configuration

Step	Command	Purpose
4	node(cfg)#reload	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.

```
SN#configure
SN(cfg)#
SN(cfg)#copy running-config startup-config
SN(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 SmartNode's running configuration on a TFTP server, where you can edit and save it. Since the SmartNode is acting as a TFTP client, it initiates all file transfer operations.

First, upload the running configuration, named *running-config*, from the SmartNode 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 SmartNode as upcoming startup configuration and store it in the persistent memory region *nvr* under the name *startup-config*. Finally, restart the SmartNode by using the **reload** command to activate the changes.

Note Consider that a customized configuration file will not modify any function of SmartWare until it has been copied to persistent memory as the new configuration file *startup-config*.

Mode: Administrator execution

Step	Command	Purpose
1	node#copy running-config tftp://node-ip-address[:port]/current-config	Uploads the current running configuration as file <i>current-config</i> to the TFTP server at address <i>node-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.

Step	Command	Purpose
3	node#copy tftp://node-ip-address/current-config nvram: startup-config	Downloads the modified configuration file <i>current-config</i> from the TFTP server at address <i>node-ip-address</i> into the persistent memory region <i>nvram:</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	node#reload	Restarts the system

Example: Modifying the running configuration offline

The following example shows how to upload the running configuration from the SmartNode 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 SmartNode's persistent memory region *nvram:* using the name *startup-config*. Finally, you must restart the SmartNode.

```
SN#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.

```
SN#copy tftp://172.16.36.80/user/current-config nvram:startup-config
Download...100%
SN#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 SmartNode flash memory region *nvram:*.

Mode: Administrator execution

Step	Command	Purpose
1	node#show nvram:	Lists the loaded configurations
2	node#erase 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 nvram:** to list all available configurations.

```
SN#show nvram:
Persistent configurations:
```

```
backup
minimal
startup-config
factory-config
```

2. Delete the configuration named *minimal* explicitly.

```
SN#erase nvram:minimal
```

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

```
SN#show nvram:
Persistent configurations:
backup
startup-config
factory-config
```


Chapter 8 **Basic system management**

Chapter contents

Introduction	96
Basic system management configuration task list	96
Managing feature license keys	97
Setting system information	98
Setting the system banner	99
Setting time and date	100
Display clock information	100
Display time since last restart	100
Configuring and starting the Web server	101
Determining and defining the active CLI version	101
Restarting the system	102
Displaying the system logs	102
Controlling command execution	103
Displaying the checksum of a configuration	104
Configuration of terminal sessions	105

Introduction

This chapter describes parameters that report basic system information to the operator or administrator, and their configuration. The following are basic SmartWare parameters that must be established when setting up a new system:

- 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:

- Checking the CRC of configuration files
- Displaying the currently running SmartWare commands
- Moving SmartWare commands into the foreground
- Setting the system banner
- Enabling the embedded web server

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 97](#))
- Setting system information (see [page 98](#))
- Setting the system banner (see [page 99](#))
- Setting time and date (see [page 100](#))
- Displaying clock information (see [page 100](#))
- Displaying time since last restart (see [page 100](#))
- Configuring and starting the web server (see [page 101](#))
- Determining and defining the active CLI version (see [page 101](#))
- Restarting the system (see [page 102](#))
- Displaying the system event log (see [page 102](#))
- Controlling command execution (see [page 103](#))
- Displaying the checksum of a configuration (see [page 104](#))
- Configuration of terminal sessions (see [page 105](#))

Managing feature license keys

Several features of the firmware require a system specific license key to be installed to enable the feature. You will receive a file containing license keys for all of your purchased features from your equipment vendor.

This section describes how to install the feature license keys on your equipment. Because license keys comprise very long strings of characters, the standard way of installing them is to download the file containing the license keys from a TFTP server to the equipment. Therefore, a TFTP server must be present in the IP network where you can store the license keys file obtained from the distributor. If no TFTP server is available, the license key can also be manually typed (or copied and pasted) in a console or Telnet window. Both procedures are described below.

Mode: Configure

Step	Command	Purpose
1	SN(cfg)#copy tftp://tftp-server/path/filename licenses:	Downloads the license key file and install the licenses.

Example: Installing license keys from a TFTP server

The following example shows the command used to install license keys, which are stored in a license file on a TFTP server.

```
SN(cfg)#copy tftp://172.16.4.3/keystore/sn1x00_120393.lic licenses:
```

Mode: Configure

Step	Command	Purpose
1	node(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.

```
SN(cfg)#install license 10011002R1Ws63yKV5v28eVmhDsVGj/JwKqIdpC4Wr1BHaNtenXUYF/
2gNLoihifacaTPLKcV+uQDG8LJis6EdW6uNk/
GxVObdEwPFJ5bTV3bIIfUZ1eUe+8c50pCCd7PSAe83Ty2c/
CnZPS1EjIrrVlJrr8VhOr1DYxkEV9evBp+tSY+y9sCeXhDwt5Xq15SAP1znTLQmym7fDakvm+z1tzswX/
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	node(cfg)#show licenses	Display all installed 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.

```
SN(cfg)#show licenses
  VPN [vpn]
  License serial number: 14343534
  Status: Active
SN(cfg)#
```

Setting system information

The system information includes the following parameters:

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

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

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.

The system name, also called the hostname, is used to uniquely identify the SmartNode 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 SmartNode the CLI prompt will be replaced with the chosen name.

Assigning explanatory location information to describe the system physical location of your SmartNode (e.g. server room, wiring closet, 3rd floor, etc.) is very supportive. This entry corresponds to the MIB II system sysLocation object.

The system provider information is used to identify the provider contact for this SmartNode 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.

The system subscriber information is used to get in touch with subscriber for this SmartNode 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.

The system supplier information is used to get in touch with the supplier for this SmartNode device, together with information on how to contact this supplier. The supplier is a company delivering SmartNode 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	node(cfg)#system contact <i>information</i>	Sets the contact information to <i>information</i>
2	node(cfg)#system hostname <i>information</i>	Sets the hostname to <i>information</i>
3	node(cfg)#system location <i>information</i>	Sets the location information to <i>information</i>
4	node(cfg)#system provider <i>information</i>	Sets the provider information to <i>information</i>
5	node(cfg)#system subscriber <i>information</i>	Sets the subscriber information to <i>information</i>
6	node(cfg)#system supplier <i>information</i>	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.

```
SN(cfg)#system contact "Bill Anybody, Phone 818 700 1504"
SN(cfg)#system hostname SN
SN(cfg)#system location Wiring Closet, 3rd Floor
SN(cfg)#system provider Best Internet Services, contact@bis.com, Phone 818 700
2340
SN(cfg)# system subscriber Mechanical Tools Inc., jsmith@mechtool.com, Phone 818
700 1402
SN(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 SmartNode via Telnet or a serial connection (see [figure 22](#)). 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:
```

Figure 22. System banner with message to operators

Mode: Configure

Step	Command	Purpose
1	node(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.

```
SN(cfg)#banner "#\n# Patton Electronics Co.\n#\n# The password of all operators has
changed\n# please contact the administrator\n#"
```

Setting time and date

All SmartNode 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	node(cfg)#clock set <i>yyyy-mm-ddThh:mm:ss</i>	Sets the system clock to <i>yyyy-mm-ddThh:mm:ss</i>

Note SmartWare includes an integrated SNTP client, which allows synchronization of time-of-day and date to a reference time server. Refer to chapter 20, “[SNTP client configuration](#)” on page 233 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.

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

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	node>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.

```
SN>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	node>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.

```
SN>show uptime
The system is up for 1 days, 23 hours, 44 minutes, 18 seconds
```

Configuring and starting the Web server

SmartNode includes an embedded web server, that can be used together with a customer-specific Java applet that must be downloaded into the persistent memory region of your SmartNode. Applets are similar to applications but they do not run as standalones. Instead, applets adhere to a set of conventions that lets them run within a Java-compatible browser. With a Java applet, custom-specific configuration tasks of SmartWare are possible using a browser instead of accessing the SmartWare CLI via Telnet or the serial console.

Without a Java applet the value of the embedded web server is limited. Contact Patton Electronics Co. for any questions about custom designed Java configuration tools for SmartWare.

Mode: Configure

Step	Command	Purpose
1	node(cfg)#webserver language {de en}	Sets the language to either German (de) or English (en).
2	node(cfg)#webserver port <i>port-number</i>	Sets the listening port number in the 1 to 65535, default port number for the web server is 80.

Example: Configuring and starting the Web server

The following example shows how to set the web server language and the listening port of your device, if you start from the configuration mode.

```
SN(cfg)#webserver language en
SN(cfg)#webserver port 80
```

Determining and defining the active CLI version

SmartWare allows having a number of CLI version installed together, whereas only one CLI version is activated. There are commands available to determine the currently running CLI version and if necessary switch to another CLI version. The idea of having several CLI version available on a system is mostly to offer reduced or enhanced command sets to users.

Mode: Configure

Step	Command	Purpose
1	node(cfg)#show version cli	Displays the currently running CLI version

Step	Command	Purpose
2	node(cfg)#cli version <i>version.revision</i>	Selects the active CLI version in the form <i>version.revision</i>

Example: Defining the desired CLI version

The following example shows how to determine the running CLI version and define CLI version 2.10 for your device, if you start from the configuration mode.

```
SN(cfg)#show version cli
CLI version : 3.00
SN(cfg)#cli version 2.10
```

Restarting the system

In case the SmartNode 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 established via the SmartNode that is to be restarted.

Mode: Administrator execution

Step	Command	Purpose
1	node#reload	Restarts the system

Example: Restarting the system

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

```
SN#reload
System configuration has been changed.
Press 'yes' to store, 'no' to drop changes : yes
Press 'yes' to restart, 'no' to cancel : yes
The system is going down
```

Displaying the system logs

The system logs contain warnings and information from the system components of SmartWare. 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.

Mode: Administrator execution

Step	Command	Purpose
1	node#show log [event]	Show event log.
2	node#show log supervisor	Show log of the system supervisor. Used For example, after an unexpectedly reboot.
3	node#show log reset	Output a list of reset reasons (with date and time).
4	node#show log boot	Displays the console and log messages captured during startup of the SmartNode.
5	node#show log login	Displays a list of succeeded and failed CLI login attempts.

Example: Displaying system logs

The following example shows how to display event log warnings and information of your device, if you start from the operator execution mode.

```
SN#show log event
2001-12-10T14:57:18 : LOGINFO      : Link down on interface internal.
2001-12-10T14:57:39 : LOGINFO      : Warm start.
2001-12-14T08:51:09 : LOGINFO      : Slot 2: Event Logging Service for ic-4brvoip -
started.
2001-12-14T08:51:09 : LOGINFO      : Slot 2: DrvPckt_Dsp_Ac48xx: DSP driver for
AC481xx created!
```

Controlling command execution

The SmartWare command shell includes a basic set of commands that allow you to control the execution of other running commands. In SmartWare, the commands **jobs** and **fg** are used for such purposes. The command **jobs** lists all running commands, and **fg** allows switching back a suspended command to the foreground. Moreover using **Ctrl-Z** suspends an active command and lets the system prompt reappear. With **Ctrl-C** the currently active command can be terminated.

Mode: Administrator execution

Step	Command	Purpose
1		Execute the first command
2	node# <Ctrl-Z>	Suspend the active command and get system prompt back
3		Execute the second command
4	node#jobs	Shows the currently running commands
5	node#fg jobid	Brings job with <i>jobid</i> back to foreground
6	node# <Ctrl-C>	Terminates the currently running command

Example: Controlling Command Execution

The following example shows how to suspend an active command, list the running commands, switch back a suspended command and terminate a currently active command on your device, if you start from the configuration mode.

```
SN>ping 172.16.36.80 1000 timeout 3
Sending 1000 ICMP echo requests to 172.16.36.80, timeout is 3 seconds:
```

```
Reply from 172.16.36.80: Time <10ms
Reply from 172.16.36.80: Time <10ms
Reply from 172.16.36.80: Time <10ms
Reply from 172.16.36.80: Time <10ms
```

Ctrl-Z suspend active command

```
% Suspended
```

System prompt reappears and is ready to execute further commands

```
SN>show ip interface
-----
Context:                router
```

Show the currently running commands

```
SN>jobs
* [run ] jobs
0 [bg ] ping
```

Bring job 0 to foreground

```
SN>fg
% Resumed [ping]
Reply from 172.16.36.80: Time <10ms
Reply from 172.16.36.80: Time <10ms
```

Ctrl-C Terminate current command

```
% Aborted (ping)
```

Displaying the checksum of a configuration

In SmartWare configuration files, e.g. startup configuration, running configuration, and user-specific configuration, contain a checksum entry. This checksum informs the user about the validity and helps distinguish configuration files on the basis of the checksum.

Mode: Administrator execution

Step	Command	Purpose
1	node#show crc filename	Displays checksum of a configuration

Example: Displaying the Checksum of a Configuration

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

```
SN#show crc nvram:test
File nvram: test:
checksum: 0xfaddc88a
```

Configuration of terminal sessions

In certain cases it may be desirable to change the settings of the current terminal session.

Mode: System

Step	Command	Purpose
1	<code>[name] (sys)#terminal height</code>	Configures the terminal height.
2	<code>[name] (sys)#[no] terminal idle-time-logout</code>	After 30 minutes without user input, a terminal session is automatically closed. If longer session periods are required (logging/debugging) this command allows to increase the session timeout, or to disable it completely.
3	<code>[name] (sys)#terminal more</code>	Enables pausing of display for commands which produce more output than the current terminal window can display at once.
4	<code>[name] (sys)#terminal width</code>	Configures the terminal width.

Chapter 9 **Radius Client Configuration**

Chapter contents

Introduction	108
The AAA component	108
General AAA Configuration	109
Radius configuration	111
Configuring Radius clients	112
Configuring the Radius server	113
Attributes in the Radius request message	113
Attributes in the Radius accept message	113
Configuring the local database accounts	114

Introduction

This chapter provides an overview of the authentication, authorization, and accounting (AAA) component in SmartWare and describes how to configure the Radius client, a subpart of the AAA component. It is important to understand how AAA works before configuring the Radius client. This chapter also describes the local database accounts configuration, which is another subpart of AAA.

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

This chapter includes the following sections:

- The AAA component
- Radius configuration (see [page 111](#))
- Configuration of the local database accounts (see [page 114](#))

The AAA component

Authentication, authorization, and accounting (AAA) 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 Remote Authentication Dial-In User Service (Radius). SmartWare supports local database and Radius AAA.

Currently, the SmartWare AAA component is used only by the login service. Authentication verifies the user by password, authorization grants access to the command line interface at administrator or operator levels. The SmartNode gets the AAA information from the local database or from one or more Radius servers. [Figure 23](#)

illustrates the authentication procedure for a user logging into a SmartNode that is configured to use Radius as authentication method.

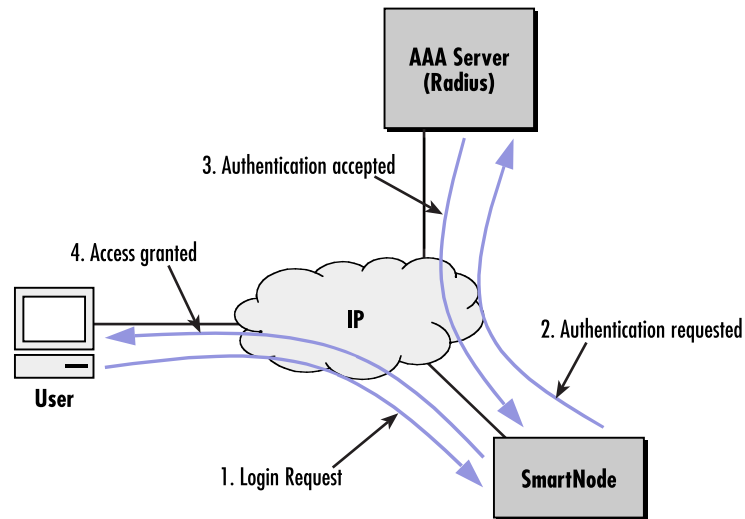


Figure 23. Authentication procedure with a Radius server

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 24 illustrates the correlation between the Telnet login and console login services.

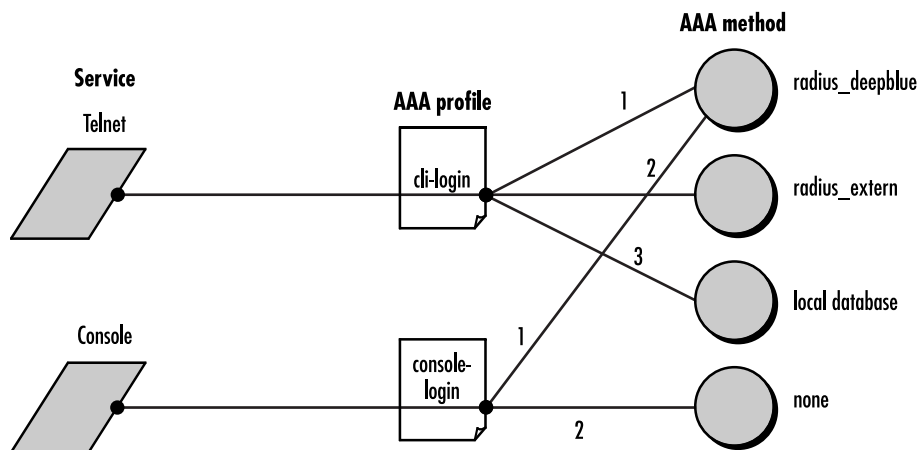


Figure 24. How to use AAA methods and AAA profiles

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 radius server *radius_deepblue*.
2. Query radius server *radius_extern*.

- Query the local database (see “Configuring the local database accounts” on page 114 for information on how to configure the local database)

If, e.g. *radius_deepblue* is not available, *radius_extern* will be queried after a timeout. But if *radius_deepblue* 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 radius server *radius_deepblue*.
- Ask predefined method *none*. This method always grants access as system operator.

If *radius_deepblue* is not available, access will be granted by the method *none*. If *radius_deepblue* rejects the login request, console access is denied. If *radius_deepblue* confirms the request, console access is granted.

Do the following to configure the AAA component.

Mode: Configure

Step	Command	Purpose
1	node(cfg)#profile authentication <i>name</i>	Creates an authentication profile with name <i>name</i> and enters profile authentication configuration mode.
2	node(pf-auth)[name]#method [<i>index</i>] { local none { Radius <i>name</i> }}	Adds an AAA method to the profile. For Radius you have to specify a name. For information on how to configure local accounts and Radius servers, refer to chapter 10, “IP context overview” on page 117. With <i>index</i> you can add a method between to others.
3		Repeat step 2 for all AAA methods you want to add
4	node(pf-auth)[name]#server-timeout <i>seconds</i>	Sets the timeout after that the next AAA method in the list is requested if no answer is received.
5	node(pf-auth)[name]#exit	Goes back to the parent configuration mode
6	node(cfg)#terminal Telnet use authentication <i>profile-name</i>	Specifies which AAA profile the Telnet login service has to use.
7	node(cfg)#terminal console use authentication <i>profile-name</i>	Specifies which AAA profile the console login service has to use.
8	node(cfg)#show profile authentication [<i>name</i>]	Displays the configured profiles

Example: Create the AAA profiles for login over Telnet and login over console, as they are shown in [figure 24](#), and use them on the Telnet login and console login services.

```
SN>enable
SN#configure
SN(cfg)#profile authentication remote-Radius
SN(pf-auth)[remote~]#method Radius Radius_deepblue
SN(pf-auth)[remote~]#method Radius Radius_extern
SN(pf-auth)[remote~]#method local
SN(pf-auth)[remote~]#server-timeout 15
```



```

SN(pf-auth)[remote-~]#exit
SN(cfg)#
SN(cfg)#profile authentication local-only
SN(pf-auth)[local-o~]#method local
SN(pf-auth)[local-o~]#method none
SN(pf-auth)[local-o~]#exit
SN(cfg)#terminal Telnet use authentication remote-Radius
SN(cfg)#terminal console use authentication local-only
SN(cfg)#show profile authentication

Authentication Profile: default
  Server-Timeout: 10
  Methods:
    local (Type=local)
    none (Type=none)

Authentication Profile: remote-Radius
  Server-Timeout: 15
  Methods:
    Radius_deepblue (Type=Radius)
    Radius_extern (Type=Radius)
    local (Type=local)

Authentication Profile: local-only
  Server-Timeout: 10
  Methods:
    local (Type=local)
    none (Type=none)

SN(cfg)#

```

**IMPORTANT**

Possible lock-out —If you delete the local and none methods from the default AAA profile, or if you create and use a profile without methods local and none, you will be unable to access your device if the network or Radius server is not available.

Note If you do not configure AAA, a default AAA profile exists containing the *AAA local* as the first AAA method and the *AAA none* as the second. The Telnet login and the console login service use this profile. If an emergency occurs, you can reload this default configuration by reloading the factory configuration as described in section “[Boot procedure](#)” on page 68.

Radius configuration

Radius is a protocol for carrying authentication, authorization, and configuration information between a network access server (NAS) that desires to authenticate its links and a shared authentication server. A NAS operates as a client of Radius. The client is responsible for passing user information to designated Radius servers and then acting on the response that is returned. Radius servers are responsible for receiving user connection requests, authenticating the user, and then returning all configuration information necessary for the client to deliver service to the user.

Transactions between the Radius client and server are authenticated through the use of a shared secret, which is never sent over the network—the same secret must thus be known to the server and the client by configuration. Using this secret as an encryption key, user passwords are sent encrypted between the client and Radius server.

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 <i>name</i>	Adds a Radius client with name <i>name</i> and enters Radius-client configuration mode
2	node(Radius)[name]#Radius-server <i>host-name</i>	Sets the hostname (or IP address) of the remote Radius server
3	node(Radius)[name]#shared-secret authentication <i>secret</i>	Sets the password shared between the Radius client (the SmartNode) and the remote Radius server.
4	node(Radius)[name]#exit	Goes back to the parent configuration mode
5	node(cfg)#show Radius-client <i>name</i>	Displays configured Radius servers

Example: Configure the Radius clients as shown in [figure 24](#).

```
SN>enable
SN#configure
SN(cfg)#Radius-client Radius_deepblue
SN(Radius)[Radius_~]#Radius-server deepblue
SN(Radius)[Radius_~]#shared-secret authentication 78f8a23b
SN(Radius)[Radius_~]#exit
SN(cfg)#Radius-client Radius_extern
SN(Radius)[Radius_~]#Radius-server 219.144.12.1
SN(Radius)[Radius_~]#shared-secret authentication dd9351e13cc335
SN(Radius)[Radius_~]#exit
SN(cfg)#
SN(cfg)#show Radius-client
Radius clients:
  Radius_deepblue
  Radius_extern
SN(cfg)#show Radius-client Radius_deepblue
AAA Radius Module: Radius_deepblue
  Authentication Shared Secret: 78f8a23b
  Timeout: 6
  Sessions:
  UDP Interface:
    Configured Server Hostname: deepblue
SN(cfg)#show Radius-client Radius_extern
AAA Radius Module: Radius_extern
  Authentication Shared Secret: dd9351e13cc335
  Timeout: 6
  Sessions:
  UDP Interface:
```

```
Configured Server Hostname: 219.144.12.1
SN(cfg)#
```

Configuring the 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 log in. Each attribute is assigned a number by RFC 2865. This section gives an overview of all such attributes that SmartWare uses. 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 request message

The SmartNode sends a Radius 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
26	Protocol	Is a vendor specific attribute that indicates the protocol with that the user wants to log on. Currently it can have the value 'console' or 'Telnet'. Thus it is possible for the Radius Server to grant access depending on whether the user wants to log on over console or Telnet

Attributes in the Radius accept message

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

Attribute number	Attribute Type	Description
6	Service-Type	If the value is set to 'administrative', the user has administrator rights on the SmartNode, otherwise operator rights
18	Reply-Message	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
27	Session-Timeout	Number of seconds the user is allowed to logged on. If the attribute is not included, the default value is infinite
28	Idle-Timeout	Number of seconds to stay in idle state before automatic logout proceeds. If the attribute is not included, the default value is 30 minutes. The command terminal idle-time-logout overwrites the value set by the attribute

Most of the attributes are standard Radius attributes and are supported by the Radius servers. You have to specify a value for each of them as it is described in your Radius server's user manual.

The attribute *Protocol* (26) is vendor specific and defined by Patton. Servers not equipped to interpret the vendor-specific information will ignore it. It is defined as follows:

```

0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|      Type      | Length |      Vendor-Id      |

```

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
      Vendor-Id (cont)           | Vendor-Type | Vendor-Length |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Vendor-String ...
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

Type: 26

Length: Length of the whole attribute including the vendor data

Vendor-Id: 5349

Vendor-Type: 16

Vendor-Length: Length of all vendor data including Vendor-Type and Vendor-Length

Vendor-String: Not null terminated String with the value *console* or *Telnet*

Configuring the local database accounts

The final step in configuring the authentication and authorization service in SmartWare is to set up local user accounts. The local database—which is queried with the AAA method *local* as described previously—can contain administrator and operator accounts. For example, to grant access to the local SmartNode if all Radius servers are down or the network is not reachable, you can create an emergency user in the local database so that you can still access the SmartNode. Perform the following steps to configure the local accounts.

Mode: Configure

Step	Command	Purpose
1	node(cfg)#[no] administrator <i>name</i> password <i>password</i>	Adds an administrator account to the local database. The no form removes an existing account
2	node(cfg)#[no] operator <i>name</i> password <i>password</i>	Adds an operator account to the local database. The no form removes an existing account
3	node(Radius)[name]#shared-secret authentication <i>secret</i>	Sets the password shared between the Radius client (the SmartNode) and the remote Radius server.
4	node(pf-auth)[name]#show accounts	Display existing accounts

Example: Create an administrator and an operator account

```

SN>enable
SN#configure
SN(cfg)#administrator meier password pencil
SN(cfg)#operator james password ""
SN(cfg)#show accounts
Administrator accounts:
  meier
Operator accounts:
  james
SN(cfg)

```

Note If you are creating an account that does not require a password, type "" to indicate that no password is needed. For example, if you were configuring an account for an operator named James that did not need a password, the entry would be:

```
SN(cfg)#operator james password ""
```


Chapter 10 **IP context overview**

Chapter contents

Introduction	118
IP context overview configuration task list	119
Planning your IP configuration	120
IP interface related information	120
Serial interface related information	121
QoS related information	121
Configuring Ethernet and serial ports	121
Creating and configuring IP interfaces	121
Configuring NAT	122
Configuring static IP routing	122
Configuring RIP	122
Configuring access control lists	123
Configuring quality of service (QoS)	123

Introduction

This chapter outlines the SmartWare *Internet protocol (IP) context* and its related components. You will get the fundamental understanding on how to set up your SmartNode 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.

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 SmartWare configuration concept by reading chapter 2, “[Configuration concepts](#)” on page 35.

The IP context in SmartWare 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 the same function as a standalone IP router, and since every context is defined by a name, the IP context is named *router* by default. This IP context can contain interface static routes, RIP parameters, NAT, QoS and access control profiles, and related SIP or H.323 gateways.

In [figure 25](#) on page 119, the IP context with all its related elements is contained within the area on the left, which has a gray fill. The right side displays the related CS context, which communicates with the IP context

via different types of gateways. Since the CS context and its related components are not the subject of this chapter, they are illustrated in [figure 25](#) with gray lines instead of black ones.

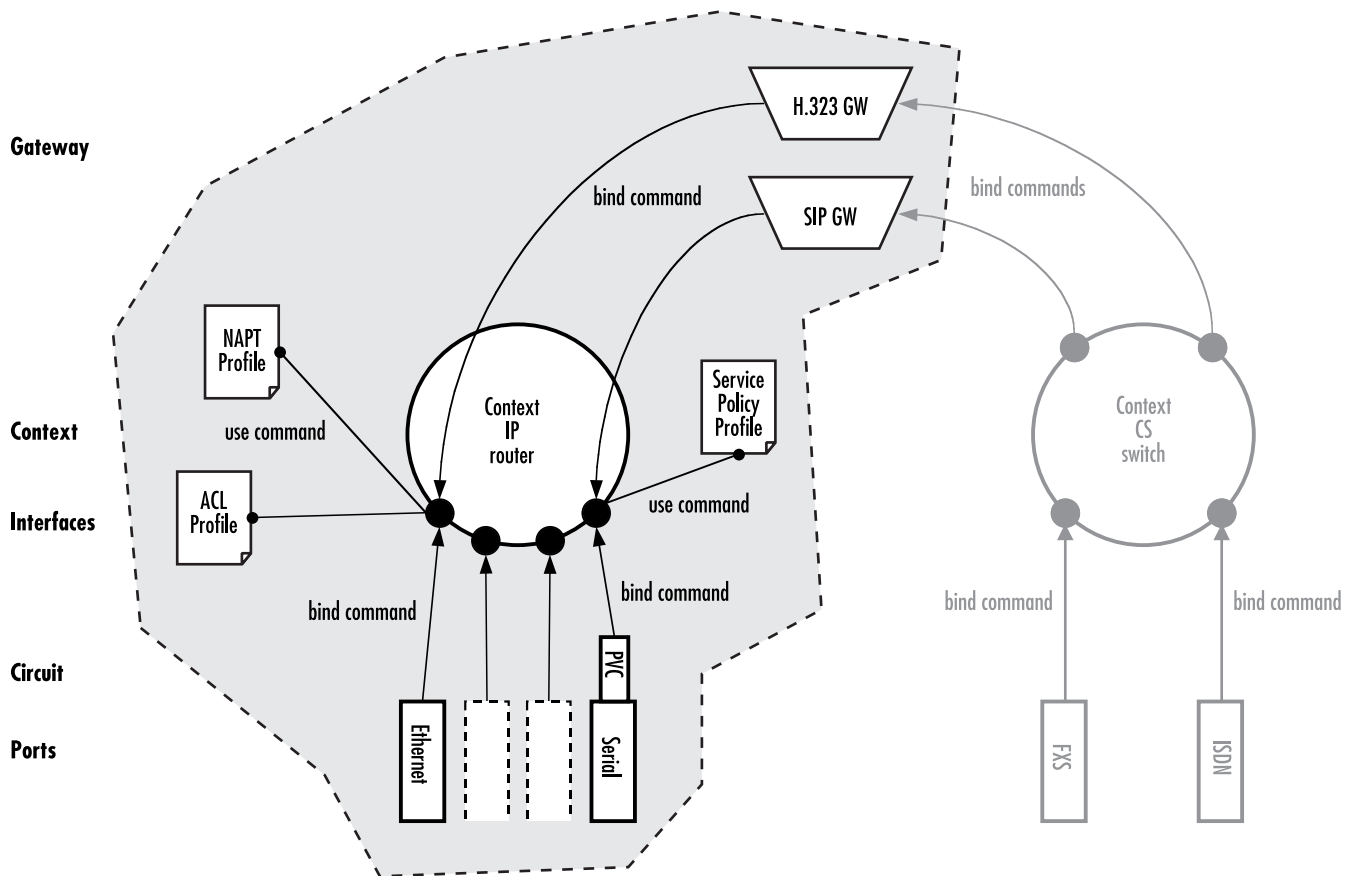


Figure 25. IP context and related elements

The IP context undertakes the task of doing all IP-related transport of data and voice packets via the logical interfaces and available gateways. In addition, using profiles—which together with the IP context pinpoint how to handle packets for specific services—enhances the possible field of application. Moreover, voice packets are transported via a voice gateway to the CS context for further processing and forwarding to the PSTN.

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.

- You can find all the information you need to configure an IP Interface in chapter 11, “[IP interface configuration](#)” on page 125.
- You can find the information regarding network address port translation (NAPT) in chapter 12, “[NAT/ NAPT configuration](#)” on page 133.
- If you need to configure a physical port, chapter 13, “[Ethernet port configuration](#)” on page 141 or chapter 15, “[Serial port configuration](#)” on page 169 may prove helpful.

- To set up the IP router contained in SmartWare, chapter 16, “[Basic IP routing configuration](#)” on page 187 and chapter 17, “[RIP configuration](#)” on page 193 give you the required information.
- For essential knowledge related to network security requirements, refer to chapter 18, “[Access control list configuration](#)” on page 203.
- If your network shall provide better service to selected network traffic, chapter 14, “[Link scheduler configuration](#)” on page 151 will help you to get in-depth knowledge about quality of service (QoS) management with SmartWare.

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 or might be optional. The following tasks use a bottom-up approach, starting from the ports, followed by the interfaces up to the services running on the SmartNode. The first tasks below shall help you obtaining the necessary overview, in view of the fact that there is always a risk getting lost in details before gaining a general understanding of the whole network.

- Planning your IP configuration (see [page 120](#))
- Configuring Ethernet and serial ports (see [page 121](#))
- Creating and configuring IP interfaces (see [page 121](#))
- Configuring NAPT (see [page 122](#))
- Configuring static IP routing (see [page 122](#))
- Configuring RIP (see [page 122](#))
- Configuring access control lists (see [page 123](#))
- Configuring quality of service (see [page 123](#))

Planning your IP configuration

The following subsections provide network connection considerations for several types of physical ports types. Patton recommends that you draw a network overview diagram displaying all neighboring IP nodes and serial connected elements. 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 SmartNode requires 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 H.323 gatekeeper
- IP addresses of the central PSTN gateway for H.323 and SIP based calls
- IP address of the central TFTP server used for configuration upload and download

Serial interface related information

The SmartNode 2300 supports the V.35 and X.21 standard for synchronous serial interfaces with speeds up to 2 Mbps. Devices that communicate over a serial interface are divided into two classes:

- Data terminal equipment (DTE)—The device at the user end of the user-to-network interface. The DTE connects to a data network via data DCE, and typically uses clocking signals generated by the DCE.
- Data communications equipment (DCE)—The device at the network end of the user-to-network interface. The DCE provides a physical connection to the network, forwards traffic, and provides a clocking signal used to synchronize data transmission between DCE and DTE devices.

The most important difference between these types of devices is that the DCE device supplies the clock signal that paces the communications on the interface.

Note The SmartNode 2300 is working as a DTE by default.

Before you connect a device to the synchronous serial port, labeled SERIAL 0/0 on SmartNode 2300, you need to check the following:

- Confirm that the device to which you are connecting to is a DCE providing a clock signal on the synchronous serial interface.
- Type of connector, male or female, required to connect at the device
- Signaling protocol required by the device must be X.21 or V.35

QoS related information

Check with your access service provider if there are any QoS related requirements, which you need to know prior to configuring SmartWare 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 Ethernet and serial ports

In SmartWare, Ethernet and serial ports represent the physical connectors on the SmartNode hardware. Since ports are closely-knit with the physical structure of a SmartNode, they cannot be created but have to be configured. The configuration of a port 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 an Ethernet port, refer to chapter 13, “Ethernet port configuration” on page 141 or for a serial port to chapter 15, “Serial port configuration” on page 169.

Creating and configuring IP interfaces

SmartWare supports one instance of the IP context, named *router*. The number and names of IP interfaces depend upon your application scenario. In SmartWare, an interface is a logical construct that provides higher-layer protocol and service information, such as layer 3 addressing. Hence interfaces are configured as part of the IP context and represent logical entities that are only usable if a physical port is bound to them.

An interface name can be any arbitrary string, but for ease of identification you should use self-explanatory names that describe the use of the interface. For example, use names like *lan* for an IP interface that connects to the LAN and *wan* for an interface that connects to the access network or WAN. Avoid names that represent the nature of the underlying physical port for logical interfaces, like *eth0* or *serial0*, to represent Ethernet port 0 or serial port 0, since IP interfaces are not strictly bound to a certain physical port. An IP interface can be moved to another physical port (from an Ethernet to a serial port, for example) while a SmartNode is operating. For that reason, it would be confusing if an interface had name like *eth0*, but was actually assigned to a serial port. So it is important to avoid naming a logical interface after a physical port. Instead, assign names to interfaces that describe their usage and not the physical connection.

Several IP-related configuration parameters are necessary to define the behavior of such an interface. The most obvious parameters are the IP address and an IP net mask that belongs to it.

For information and examples on how to create and configure an IP interface, refer to chapter 11, “[IP interface configuration](#)” on page 125.

Configuring NAPT

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. NAPT enables small offices to save money by requiring only one official outside IP address to connect several hosts via a SmartNode to the access network. Moreover, NAPT provides additional security, because the IP addresses of hosts attached via the SmartNode are invisible to the external world. You can configure NAPT by creating a profile that is afterwards used on an explicit IP interface. In SmartWare terminology, an IP interface *uses* a NAPT profile, as shown in [figure 25](#) on page 119.

For information and examples on how to configure NAPT refer to chapter 12, “[NAT/NAPT configuration](#)” on page 133.

Configuring static IP routing

SmartWare allows to define static routing entries, which are table mappings established by the network administrator prior to the beginning of routing. These mappings do not change unless the network administrator alters 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.

For information and examples on how to configure static IP routing, refer to chapter 16, “[Basic IP routing configuration](#)” on page 187.

Configuring RIP

The Routing Information Protocol (RIP) is a distance-vector protocol that uses hop count as its metric. RIP is widely used for routing traffic in the global Internet and is an interior gateway protocol (IGP), which means that it performs routing within a single autonomous system.

RIP sends routing-update messages at regular intervals and also when the network topology changes. When a router receives a routing update that includes changes to an entry, it updates its routing table to reflect the new route. The metric value for the path is increased by one, and the sender is indicated as the next hop. RIP routers maintain only the best route (the route with the lowest metric value) to a destination. After updating its routing table, the router immediately begins transmitting routing updates to inform other network routers of the change. These updates are sent independently of the regularly scheduled updates that RIP routers send.

RIP uses a single routing metric (hop count) to measure the distance between the source and a destination network. Each hop in a path from source to destination is assigned a hop-count value, which is typically 1. When a router receives a routing update that contains a new or changed destination-network entry, the router adds one to the metric value indicated in the update and enters the network in the routing table. The IP address of the sender is used as the next hop.

RIP prevents routing loops from continuing indefinitely by implementing a limit on the number of hops allowed in a path from the source to a destination. The maximum number of hops in a path is 15. If a router receives a routing update that contains a new or changed entry, and if increasing the metric value by one causes the metric to be infinity (i.e. 16), the network destination is considered unreachable.

For information and examples on how to configure Routing Information Protocol (RIP) refer to chapter 17, “[RIP configuration](#)” on page 193.

Configuring access control lists

Packet filtering helps to control packet movement through the network. Such control can help to limit network traffic and to restrict network use by certain users or devices. To permit or deny packets from crossing specified interfaces, SmartWare provides access control lists.

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 SmartNode. SmartWare tests packets against the conditions in an access list one by one. The first match determines whether SmartWare accepts or rejects the packet. Because SmartWare 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 18, “[Access control list configuration](#)” on page 203.

Configuring quality of service (QoS)

In SmartWare, the link scheduler enables the definition of QoS profiles for network traffic on a certain interface, as shown in [figure 25](#) on page 119. QoS refers to the ability of a network to provide improved service to selected network traffic over various underlying technologies including Frame Relay, Ethernet and 802.x type networks, and IP-routed networks. In particular, QoS features provide improved and more predictable network service by providing the following services:

- Supporting dedicated bandwidth
- Improving loss characteristics
- Avoiding and managing network congestion
- Shaping network traffic
- Setting traffic priorities across the network

SmartWare QoS features described in chapter 14, “[Link scheduler configuration](#)” on page 151 address these diverse and common needs.

Chapter 11 IP interface configuration

Chapter contents

Introduction	126
Software IP interface configuration task list	126
Creating an IP interface	126
Deleting an IP interface	127
Setting the IP address and netmask	127
ICMP message processing	128
ICMP redirect messages	128
Router advertisement broadcast message	129
Defining the MTU and MSS of the interface	129
Configuring an interface as a point-to-point link	130
Displaying IP interface information	131
Testing connections with the ping command	131
Traceroute	132
Examples	132
Deleting an IP interface	132

Introduction

This chapter provides a general overview of SmartNode interfaces and describes the tasks involved in their configuration.

Within SmartWare, 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 of the advanced features offered by SmartWare. 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, SDSL or Frame Relay ports according to the appropriate transport network layer.

Software IP interface configuration task list

To configure interfaces, perform the tasks in the following sections:

- Creating an IP interface (see [page 126](#))
- Deleting an IP interface (see [page 127](#))
- Setting the IP address and netmask (see [page 127](#))
- ICMP message processing (see [page 128](#))
- ICMP redirect messages (see [page 128](#))
- Router advertisement broadcast message (see [page 129](#))
- Defining the MTU of the interface (see [page 129](#))
- Configuring an interface as a point-to-point link (see [page 130](#))
- Displaying IP interface information (see [page 131](#))
- Testing connections with the **ping** command (see [page 131](#))

Creating an IP interface

Interface names can be any arbitrary string. Use self-explanatory names for your interfaces, which reflect their usage.

Mode: Context IP

Step	Command	Purpose
1	node(ctx-ip)[router]#interface <i>name</i>	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.
2	node(if-ip)[name]#	You are now in the interface configuration mode, where you can enter specific configuration parameters for the IP interface <i>name</i> .

Example: Create IP interfaces

The procedure illustrated below assumes that you would like to create an IP interface named *lan*. Use the following commands in administrator configuration mode.

```
SN>enable
SN#configure
SN(cfg)#context ip router
SN(ctx-ip)[router]#interface lan
SN(if-ip)[lan]#
```

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.

Mode: Context IP

Step	Command	Purpose
1	node(ctx-ip)[router]#no interface <i>name</i>	Deletes the existing interfaces <i>name</i>

Example: Delete IP interfaces

The procedure below assumes that you would like to delete an IP interface named *external*. Use the following commands in IP context configuration mode.

List the existing interfaces:

```
SN(ctx-ip)[router]#interface <?>
<interface>          New interface
lan                   Existing interface
wan                   Existing interface
external              Existing interface
internal              Existing interface
```

Delete the interfaces named *eth3* with the **no interface** command:

```
SN(ctx-ip)[router]#no interface external
```

List the interfaces again to check if the appropriate interface was deleted:

```
SN(ctx-ip)[router]#interface <?>
<interface>          New interface
lan                   Existing interface
wan                   Existing interface
internal              Existing interface
```

Setting the IP address and netmask

Each IP interface needs its explicit IP address and an appropriate net mask to be set. You can use the **ipaddress** interface configuration command to perform the following tasks:

- Set the IP address to *ip-address*

- Set the network mask to *netmask*
- Enable IP processing for the IP interface *name* without assigning an explicit IP address

The `ipaddress` command offers the following options:

unnumbered	Enables IP processing on an interface without assigning an explicit IP address to the interface.
<i>ip-address</i>	Specifies the IP address of the subscriber in the form A.B.C.D.
<i>netmask</i>	Specifies the network mask in the form A.B.C.D. A network mask of at least 24 bits must be entered; i.e. a mask in the range 255.255.255.0 through 255.255.255.255.
dhcp	Enables the DHCP client on this interface. For more information on DHCP-client configuration refer to chapter 21, “ DHCP configuration ” on page 245.

Mode: Context IP. This command also places you in interface configuration mode.

Step	Command	Purpose
1	<code>node(ctx-ip)[router]#interface name</code>	Selects the existing interface <i>name</i> , which shall be configured
2	<code>node(if-ip)[name]# ipaddress {unnumbered (ip-address netmask) dhcp}</code>	Sets the IP address <i>ip-address</i> and netmask <i>netmask</i> for interface <i>name</i>

Example: Configure IP interface address and netmask

To set the IP address to *192.168.1.3* and net mask to *255.255.255.0* for the IP interface *lan*, use the following commands in IP context configuration mode.

```
SN(ctx-ip)[router]#interface lan
SN(if-ip)[lan]#ipaddress 192.168.1.3 255.255.255.0
```

ICMP message processing

The IP suite offers a number of services that control and manage IP connections. The Internet Control Message Protocol (ICMP) provides many of these services. Routers send ICMP messages to hosts or other routers when a problem is discovered with the Internet header. For detailed information on ICMP, see RFC 792. SmartWare supports the following ICMP message processing features:

- ICMP redirect messages
- Router advertisement broadcast message

ICMP redirect messages

Routes are sometimes less than optimal. For example, the router may be forced to resend a packet through the same interface on which it was received. In this case, the SmartWare application software sends an ICMP redirect message to the originator of the packet telling the originator that the router is on a subnet directly connected to the receiving device, and that it must forward the packet to another system on the same subnet. The software sends an ICMP redirect message to the originator of the packet because the originating host presumably could have sent that packet to the next hop without involving this device at all. The redirect message instructs the sender to remove the receiving device from the route and substitute a specified device representing a more direct path. This feature is enabled by default.

SmartWare ICMP message processing offers two options for host route redirects:

- `accept`—accepts ICMP redirect messages
- `send`—sends ICMP redirect messages

Mode: Interface

Step	Command	Purpose
1	<code>node(ctx-ip)[router]#interface name</code>	Selects the interface <i>name</i> for ICMP message processing configuration
2	<code>node(if-ip)[name]#icmp redirect { accept send}</code>	Enables to send or accept ICMP redirect messages

Example: ICMP redirect messages

The following example shows how to configure ICMP messages processing to accept ICMP redirect messages on the IP interface *lan*. Use the following commands in IP context configuration mode.

```
SN(ctx-ip)[router]#interface lan
SN(if-ip)[lan]#icmp redirect accept
```

Router advertisement broadcast message

This message configures the behavior of the router when receiving an ICMP router solicitation message, and determines if the router shall send periodic ICMP router advertisement messages or not.

By default, ICMP router advertisement messages are sent, either as a reply to ICMP router solicitation messages or periodically. If the feature is disabled, ICMP router advertisement messages are not sent in any case, neither as a reply to ICMP router solicitation messages nor periodically.

Mode: Interface

Step	Command	Purpose
1	<code>node(ctx-ip)[router]#interface name</code>	Selects the interface <i>name</i> for ICMP message processing configuration
2	<code>node(if-ip)[name]# icmp router-discovery</code>	Enables to send router advertisement broadcast messages

Example: Router advertisement broadcast message

The following example shows how to enable sending router advertisement broadcast messages on IP interface *lan*. Use the following commands in IP context configuration mode.*

```
SN(ctx-ip)[router]#interface lan
SN(if-ip)[lan]#icmp router-discovery
```

Defining the MTU and MSS of the interface

All interfaces have a default MTU packet size. You can adjust the IP MTU size so that the SmartWare application software will fragment any IP packet that exceeds the MTU set for an interface. The default MTU packet size is set to 1500 for an interface. In cases where fragmentation is not allowed along the IP connection, forcing a reduction of the MSS (*maximum segment size*) is the only viable solution.

Note All devices on a physical medium must have the same protocol MTU in order to operate accurately.

Procedure: To set the MTU packet size or the MSS to *size* on the interface *name*

Mode: Interface

Step	Command	Purpose
1	<code>node(ctx-ip)[router]#interface name</code>	Selects the interface <i>name</i> for ICMP message processing configuration
2	<code>node(if-ip)[name]#mtu size</code>	Sets the IP MTU packet size to <i>size</i> of the interface <i>name</i> . The MTU packet size value must be in the range from 48 to 1500.
Step 3 (optional)	<code>node(if-ip)[name]#tcp adjust-mss { rx tx } { mtu mss }</code>	Limits to the MSS (Maximum Segment Size) in TCP SYN packets to <i>mss</i> or to MTU (Maximum Transmit Unit) - 40 Bytes, if 'mtu' is used. 'rx' applies to packets which arrive inbound at this IP interface, 'tx' to packets which leave outbound of this IP interface. It is recommended to use 'mtu' inbound and outbound.

Example: Defining the MTU of the interface

The following example shows how to define the MTU of the IP interface *lan* to 1000 and to adjust the MSS in both directions to MTU-40. Use the following commands in IP context configuration mode.

```
SN(ctx-ip)[router]#interface lan
SN(if-ip)[lan]#mtu 1000
SN(if-ip)[lan]#tcp adjust-mss rx mtu
SN(if-ip)[lan]#tcp adjust-mss tx mtu
```

Configuring an interface as a point-to-point link

A point-to-point network joins a single pair of routers. It is in particular used for interfaces, which have a binding to a Frame Relay PVC.

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg)#context ip router</code>	Selects the IP router context
2	<code>node(ctx-ip)[router]#interface name</code>	Selects the defined interface <i>name</i> for configuration
3	<code>node(if-ip)[name]#point-to-point</code>	Configures the interface ifname as point-to-point link

Example: Configuring an interface as a point-to-point link

The following example shows how to define the interface *lan* as point-to-point link. Use the following commands in configuration mode.

```
SN(cfg)#context ip router
SN(ctx-ip)[router]#interface lan
SN(if-ip)[lan]#point-to-point
```

Displaying IP interface information

SmartWare contains the **show ip interface** command, which displays IP information for all interfaces. The command is available in operator execution mode or in any of the administrator execution modes.

Mode: Operator execution or any administrator execution

Step	Command	Purpose
1	node>show ip interface	Displays the IP information for all interfaces

Example: Displaying IP interface information

The following example shows how to display the IP information for all interfaces by using the **show ip interface** command from operator execution mode.

```
SN>show ip interface
-----
Context:          router
Name:             lan
IP Address:       172.16.40.77 255.255.0.0
MTU:              1500
ICMP router-discovery: enabled
ICMP redirect:    send only
State:            OPENED
Binding:          ethernet 0 0 0/ethernet/ip

-----
Context:          router
Name:             wan
IP Address:       172.17.100.210 255.255.255.0
MTU:              1500
ICMP router-discovery: enabled
ICMP redirect:    send only
State:            CLOSED
Binding:          ethernet 0 0 1/ethernet/ip
```

Testing connections with the ping command

As an aid to diagnosing basic network connectivity, many network protocols support an echo protocol. The protocol involves sending a special 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: Either operator or administrator execution

Step	Command	Purpose
1	node>ping ip-address	Sends ICMP ECHO_REQUEST packets to network hosts at IP address <i>ip-address</i>

When using **ping** for fault isolation, you should first run it on the respective SmartNode interface to verify that the local LAN or WAN interface is up and running. Then, you should “ping” hosts and gateways further away. Round-trip times and packet loss statistics are computed. 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 five ICMP echo requests packets have been sent and received, a brief summary 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 execution mode.

```
SN>ping 172.16.1.10
Sending 5 ICMP echo requests to 172.16.1.10, timeout is 1 seconds:
Reply from 172.16.1.10: Time <10ms
Reply from 172.16.1.10: Time <10ms
Reply from 172.16.1.10: Time <10ms
Reply from 172.16.1.10: Time <10ms
Reply from 172.16.1.10: Time <10ms
Ping statistics for 172.16.1.10:
    Packets: Sent 5, Received 5, Lost 0 (0% loss),
    RTT:      Minimum <10ms, Maximum <10ms, Average <10ms
```

Traceroute

This procedure describes how to print the route (list of hops) packets take to the network host.

Mode: Either operator or administrator execution

Step	Command	Purpose
1	node>traceroute <i>ip-address</i>	Prints the route (list of hops) packets take to network host.

Examples

Deleting an IP interface

The following example shows how to delete an IP interface named *wan*.

List the existing interfaces in the IP context:

```
SN(ctx-ip)[router]#interface <?>
<interface>          New interface
lan                  Existing interface
wan                  Existing interface
```

Delete the interface *wan* by using the **no** form of the **interface** command.

```
SN(ctx-ip)[router]#no interface wan
```

List the interfaces again to make sure that interface *wan* no longer exists:

```
SN(ctx-ip)[router]#interface <?>
<interface>          New interface
lan                  Existing interface
```

Chapter 12 **NAT/NAPT configuration**

Chapter contents

Introduction	134
Dynamic NAPT	134
Static NAPT	135
Dynamic NAT	136
Static NAT	136
NAPT traversal	137
NAT/NAPT configuration task list	137
Creating a NAPT profile	137
Activate NAT/NAPT	138
Displaying NAT/NAPT configuration information	139

Introduction

This chapter provides a general overview of Network Address (Port) Translation and describes the tasks involved in its configuration.

The two most compelling problems facing the IP Internet are IP address depletion and scaling in routing. Long-term and short-term solutions to these problems are being developed. The short-term solution is CIDR (Classless Inter Domain Routing). The long-term solutions consist of various proposals for new internet protocols with larger addresses.

Until the long-term solutions are ready, an easy way to hold down the demand for IP addresses is through address reuse. This solution takes advantage of the fact that a very small percentage of hosts in a stub domain are communicating outside of the domain at any given time (a stub domain is a domain, such as a corporate network, that only handles traffic originated or destined to hosts in the domain). Indeed, many (if not most) hosts never communicate outside of their stub domain. Because of this, only a subset of the IP addresses inside a stub domain need to be translated into IP addresses that are globally unique when outside communications is required.

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.

SmartWare 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.

Note Only the NAT/NAPT component handles global NAPT addresses. Other components of the SmartNode (e.g. the H.323 gateway) are not accessible via these addresses.

Figure 26 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.

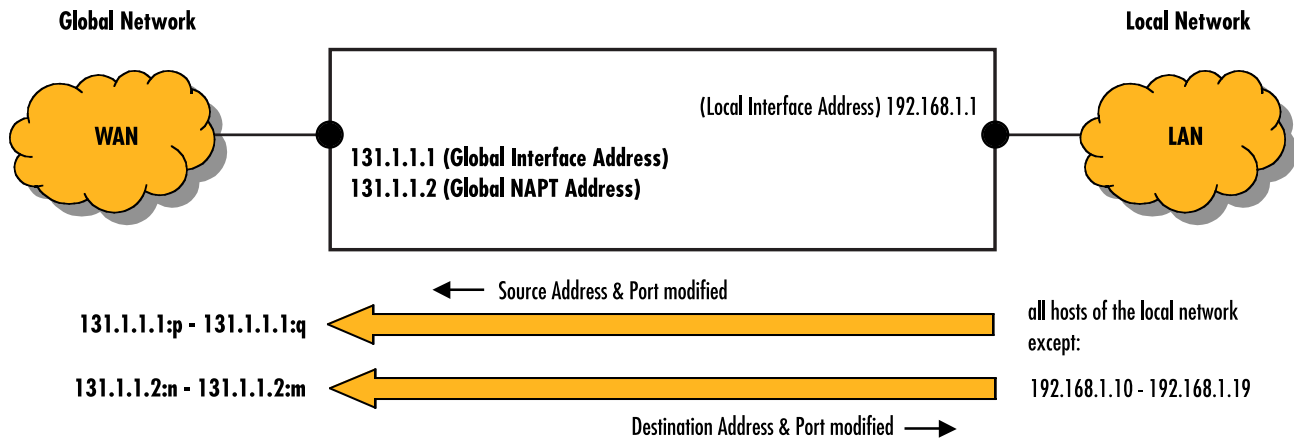


Figure 26. Dynamic NAPT

Static NAPT

Dynamic NAPT does not permit hosts on the global network to access hosts on the local network. 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.

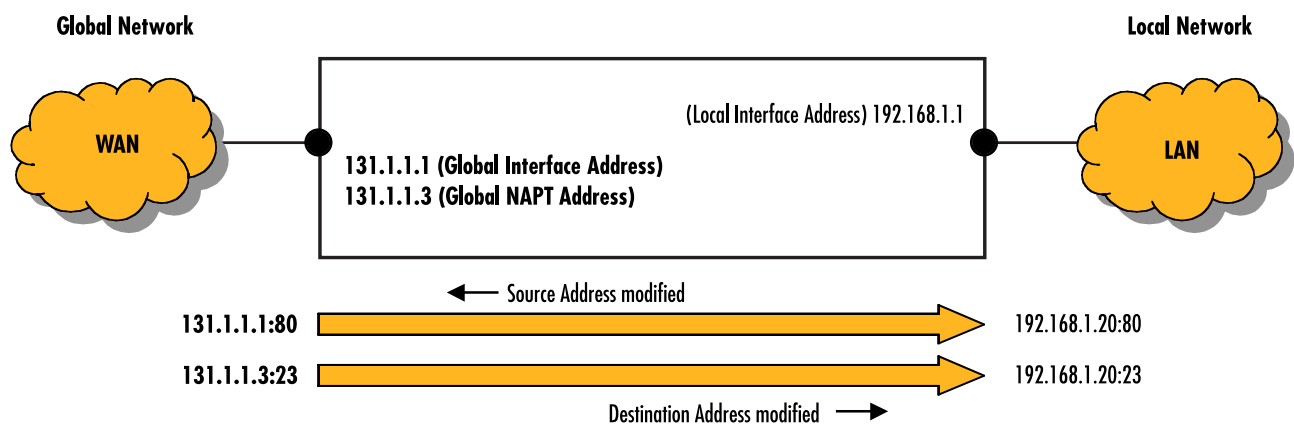


Figure 27. Static NAPT

Note Be careful when mapping ports the SmartNode uses itself (e.g. Telnet, TFTP) because the SmartNode 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 137.

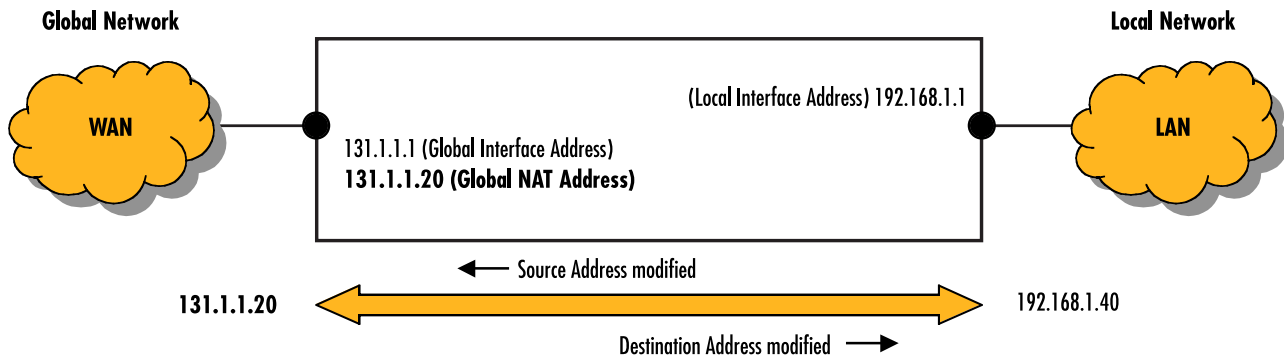


Figure 28. Dynamic NAT

Static NAT

Dynamic NAT does not permit hosts on the global network to access hosts on the local network. 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 SmartNode 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 137.

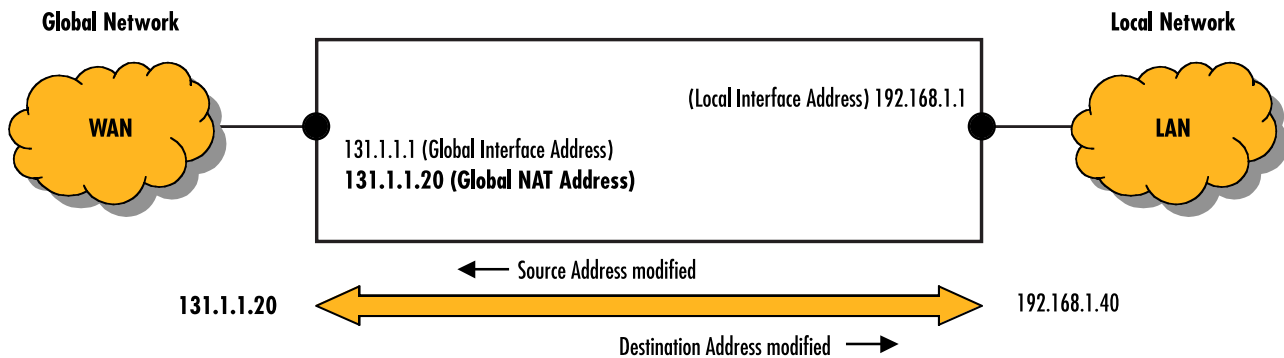


Figure 29. Static NAT

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, H.323, SIP), do not easily traverse a NAPT.

The SmartWare 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 H.323 and SIP cannot traverse the SmartWare 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 137](#))
- Activating NAT/NAPT (see [page 137](#))
- Displaying NAT/NAPT configuration information (see [page 139](#))

Creating a NAPT profile

A NAPT profile defines the behavior of the NAT/NAPT component, 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

	Command	Purpose
Step 1	<code>node(cfg)#profile napt name</code>	Creates the NAPT profile <i>name</i> and activates the basic behavior of the Dynamic NAPT
Step 2 (optional)	<code>node(pf-napt)[name]#range local-ip-range-start local-ip-range-stop global-ip</code>	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.

	Command	Purpose
Step 3 (optional)	<code>node(pf-napt)[name]#static { udp tcp } local-ip local-port [global-ip] [global-port]</code>	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. (max. 20 UDP and 20 TCP entries)
Step 4 (optional)	<code>node(pf-napt)[name]#range local-ip-range-start local-ip-range-stop global-ip-start global-ip-stop</code>	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.
Step 5 (optional)	<code>node(pf-napt)[name]#static local-ip global-ip</code>	Creates a Static NAT entry: <i>local-ip</i> is mapped to <i>global-ip</i> . (max. 20 entries)

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 134.

```
SN(cfg)#profile napt access
SN(pf-napt)[access]#range 192.168.1.10 192.168.1.19 131.1.1.2
SN(pf-napt)[access]#static tcp 192.168.1.20 80
SN(pf-napt)[access]#static tcp 192.168.1.20 23 131.1.1.3
SN(pf-napt)[access]#range 192.168.1.30 192.168.1.39 131.1.1.10 131.1.1.15
SN(pf-napt)[access]#static 192.168.1.40 131.1.1.20
```

Activate NAT/NAPT

To activate a NAT/NAPT component, bind its NAPT profile to an IP interface. This binding identifies the global interface of the respective NAT/NAPT component. All other IP interfaces are local relative to this NAT/NAPT.

Note If both a NAPT profile and an ACL profile are bound to the same IP interface, the ACL (Access Control List) acts on the local side of the NAT/NAPT component.

Procedure: To activate a NAT/NAPT component

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg)#context ip router</code>	Selects the IP router context
2	<code>node(ctx-ip)[router]#interface name</code>	The NAPT profile shall be used on the interface <i>name</i>
3	<code>node(if-ip)[name]#use profile napt profile</code>	Defines that the NAPT profile <i>profile</i> shall be used on the interface <i>name</i>

Example: Configuring NAPT Interface

The following example shows how to activate a NAT/NAPT component with the NAPT profile *access* on the IP interface *lan*.

```
SN(cfg)#context ip router
SN(ctx-ip)[router]#interface lan
SN(if-ip)[lan]#use profile napt access
```

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

	Command	Purpose
Step 1	<code>node(cfg)#show profile napt</code>	Displays the available NAPT profiles
Step 2	<code>node(cfg)#show profile napt name</code> or <code>node(cfg)#show napt interface name</code>	Displays the NAPT profile <i>name</i> or Displays the NAPT profile bound to the IP interface <i>name</i>

Example: Display NAT/NAPT Configuration Information

```
SN(cfg)#show profile napt
NAPT profiles:
-----
access

SN(cfg)#show profile napt access
NAPT profile access:
-----
ICMP default server: (none)

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 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  131.1.1.2

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 13 **Ethernet port configuration**

Chapter contents

Introduction	142
Ethernet port configuration task list	142
Entering the Ethernet port configuration mode	143
Configuring medium for an Ethernet port	143
Configuring Ethernet encapsulation type for an Ethernet port	144
Binding an Ethernet port to an IP interface	144
Selecting the frame format for an Ethernet port	145
Configuring layer 2 CoS to service-class mapping for an Ethernet port	146
Adding a receive mapping table entry	147
Adding a transmit mapping table entry	147
Closing an Ethernet port	148

Introduction

This chapter provides an overview of Ethernet ports and describes the tasks involved in configuring Ethernet ports through the SmartWare.

For SmartNode Series devices, the term Ethernet refers to the family of local area network (LAN) or wide area network (WAN) implementations that include two principal categories.

- Ethernet and IEEE — 802.3 LAN/WAN specifications that operate at 10 Mbps over twisted-pair and coaxial cable.
- 100 Mbps Ethernet — LAN/WAN specification, also known as Fast Ethernet that operates at 100 Mbps over twisted-pair cable.

The information in this chapter applies to all Ethernet ports on the system, including the Ethernet management port.

Ethernet port configuration task list

To configure Ethernet ports, perform the tasks described in the following sections. Most of the task 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 143](#))
- Configuring medium for an Ethernet port (see [page 143](#))
- Configuring Ethernet encapsulation type for an Ethernet port (see [page 144](#))
- Binding an Ethernet port to an IP interface (see [page 144](#))
- Selecting the frame format for an Ethernet port (see [page 145](#))
- Configuring layer 2 CoS to service-class mapping for an Ethernet port (advanced) (see [page 146](#))
- Closing an Ethernet port (see [page 148](#))

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 as show in table 6.

Table 6. Permanent built-in interface slot and port mapping for SmartNode 1x00, 2x00, and 4xxx Series

Device Type	Interface Type	Slot	Port	Interface
SmartNode 1x00	Ethernet	0	0	ETH 0
			1	ETH 1
	ISDN	0	0	BRI 0
			1	BRI 1
SmartNode 2x00	Ethernet	0	0	ETH 0/0
			1	ETH 0/1
	Serial (SN2300 only)	0	0	SERIAL 0/0
SmartNode 4xxx	Ethernet	0	0	ETH 0/0
			1	ETH 0/1 ^a
	FXS	0	0	FXS 0/0
		
			7 (max)	FXS 0/7

a. Available on 45xx only

Since a port must be configured unambiguously, choose the appropriate expansion slot and port number. The number and type of available ports depend upon your SmartNode model, and also on the interface card fit for SmartNode 2000 series devices. All permanent on-board interfaces of a SmartNode are described as being on slot 0.

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. Supported command options are:

- **10**—for 10 Mbps
- **100**—for 100 Mbps
- **auto**—for auto-sense the port speed
- **half**—for half-duplex
- **full**—for full-duplex

This procedure describes how to configure the medium for the Ethernet port on *slot* and *port*

Mode: Configure

Step	Command	Purpose
1	node(cfg)#port ethernet <i>slot port</i>	Enters Ethernet port configuration mode for the interface on <i>slot</i> and <i>port</i> .
2	node(prt-eth)[<i>slot/port</i>]#medium (10 100 auto) (half full)	Configures the interface on <i>slot</i> and <i>port</i> to medium according to the selected option.

Note The following restrictions apply:

- SN1x00: Both Ethernet ports support 10 Mbps half-duplex
- SN2300: Ethernet port 0/1 only supports 10 Mbps half-duplex, other settings are ignored
- SN4xxx: Ethernet port 0/1 supports 10 Mbps half-duplex.

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 of a SmartNode 4524 device.

```
SN(cfg)#port ethernet 0 0
SN(prt-eth)[0/0]#medium auto
```

Configuring Ethernet encapsulation type for an Ethernet port

This procedure describes how to configure the encapsulation type to IP for the Ethernet port on *slot* and *port*.

Mode: Configure

Step	Command	Purpose
1	node(cfg)#port ethernet <i>slot port</i>	Enters Ethernet port configuration mode for the interface on <i>slot</i> and <i>port</i> .
2	node(prt-eth)[<i>slot/port</i>]#encapsulation ip	Configures the encapsulation type to IP.

Example: Configuring Ethernet encapsulation type for an Ethernet port

The following example shows how to configure the encapsulation type to IP for the Ethernet port on slot 0 and port 0 of a SmartNode 1000, 2000, or 4000 series device.

```
SN(cfg)#port ethernet 0 0
SN(prt-eth)[0/0]#encapsulation ip
```

Binding an Ethernet port to an IP interface

You must bind the Ethernet port to an existing IP interface. When executing the **bind** command, the requested interface must exist. If no IP context is given, the system attaches the interface to the default IP context known as *router*.

Figure 30 shows the logical binding of the Ethernet port at slot 0 on port 0 to the IP interface *lan* which is defined in the IP context router.

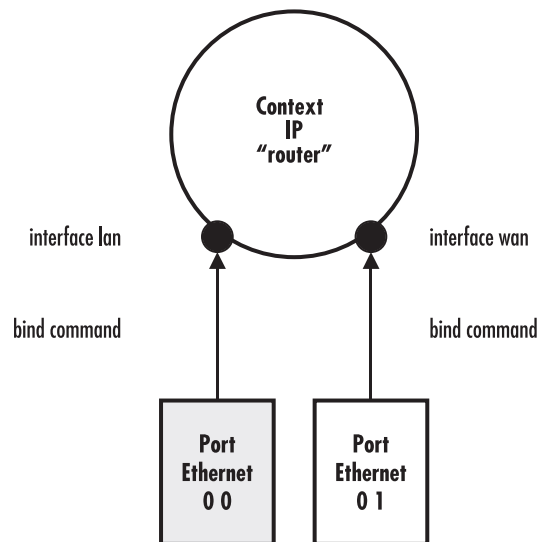


Figure 30. Binding of an Ethernet port to an IP interface

This procedure describes how to bind the Ethernet port to an already existing IP interface

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	<code>node(prt-eth)[slot/port]#bind interface name router</code>	Binds the Ethernet port to the already existing IP interface <i>if-name</i>

Example: Binding an Ethernet port to an IP interface

The following example shows how to bind the Ethernet port on slot 0 and port 0 of a SmartNode 1000, 2000, or 4000 series device to an already existing IP interface *lan*.

```
SN(cfg)#port ethernet 0 0
SN(prt-eth)[0/0]#bind interface lan router
```

Selecting the frame format for an Ethernet port

The frame format defines the logical grouping of information sent as a data link layer unit over a transmission medium. Depending on the components receiving data sent from a SmartNode via an Ethernet connection, the frame format has to be specified. The command `frame-format` allows you to set the sending of either IEEE 802.3 or IEEE 802.1Q frames. Supported command options are:

- **dot1q**—Sends VLAN-tagged IEEE 802.1Q frames used for virtual LANs
- **standard**—Sends standard IEEE 802.3 Ethernet frames

By default, the frame format is set to standard, representing IEEE 802.3.

This procedure describes how to change the frame format of the Ethernet port on *slot* and *port*.

Mode: Configure

Step	Command	Purpose
1	node(cfg)#port ethernet <i>slot port</i>	Enters Ethernet port configuration mode for the interface on <i>slot</i> and <i>port</i>
2	node(prt-eth)[<i>slot/port</i>]#frame-format {standard dot1q}	Selects to send standard IEEE 802.3 or VLAN-tagged IEEE 802.1Q frames

Example: Binding an Ethernet port to an IP interface

The following example shows how to bind the Ethernet port on slot 0 and port 0 of a SmartNode 1000, 2000, or 4000 series device to send tagged IEEE 802.1Q frames.

```
SN(cfg)#port ethernet 0 0
SN(prt-eth)[0/0]#frame-format dot1q
```

Configuring layer 2 CoS to service-class mapping for an Ethernet port

To enable to transport real-time and delay sensitive services such as VoIP traffic across the network, the firmware application software supports the delivery of Quality of Service (QoS) information in the ToS (Type of Service) field. This is an eight-bit field, the second field in the IP header packet. To define the Class of Service (CoS) to service class mapping, the **cos** command is used, with one of the following arguments:

- **default**—Default service class when no Layer 2 CoS present
- **rx-map**—Receive mapping table - Layer 2 CoS to service class mapping
- **tx-map**—Transmit mapping table - Service class to Layer 2 CoS mapping

This procedure describes how to change layer 2 CoS to service class mapping.

Mode: Configure

Step	Command	Purpose
1	node(cfg)#port ethernet <i>slot port</i>	Enters Ethernet port configuration mode for the interface on <i>slot</i> and <i>port</i>
2	node(prt-eth)[<i>slot/port</i>]#map cos <i>layer 2 class of service value</i> to <i>traffic class name</i>	Selects the layer 2 CoS to traffic-class mapping. The traffic class name can be freely chosen.

If the frame format is set to standard, the **cos default** command value defines which class of service to use for the data traffic.

The **cos rx-map** and **cos tx-map** commands above need service class mapping table entries, which has to be entered as additional command argument. The command syntax is:

- **cos rx-map**—layer 2 class of service value as service class value
- **cos tx-map**—service class value as layer 2 class of service value

Do the following to configure the class of service map:

1. Configure the class of service map table for the outgoing data traffic. Every provided service can be mapped to a Class of Service.
2. Configure the class of service map table for the incoming data traffic. Every received Class of Service can be assigned to a service type.

Adding a receive mapping table entry

The receive mapping table defines the conversion of receiving Layer 2 CoS to service class value into a firmware-specific service class value. Each conversion is stored as a mapping table entry, so the receive mapping table consists of several mapping table entries.

This procedure describes how to add a receive mapping table entry.

Mode: Configure

Step	Command	Purpose
1	node(cfg)#port ethernet <i>slot port</i>	Enters Ethernet port configuration mode for the interface on <i>slot</i> and <i>port</i> .
2	node(prt-eth)[slot/port]#cos rx-map <i>layer 2 class of service value as service class value</i>	Adds a receive mapping table entry, which converts a <i>layer 2 class of service</i> into a <i>service class value</i> .

Example: Adding a receive mapping table entry

The following example shows how to add a receive mapping table entry, which converts a layer 2 class of service value of 2 into a service class value of 4 for the Ethernet port on slot 0 and port 0 of a SmartNode.

```
SN(cfg)#port ethernet 0 0
SN(prt-eth)[0/0]#cos rx-map 2 as 4
```

Adding a transmit mapping table entry

The transmit mapping table defines the conversion of transmitting firmware-specific service class value into a Layer 2 CoS to service class value. Each conversion is stored as a mapping table entry, so the transmitting mapping table consists of several mapping table entries.

This procedure describes how to add a transmit mapping table entry.

Mode: Configure

Step	Command	Purpose
1	node(cfg)#port ethernet <i>slot port</i>	Enters Ethernet port configuration mode for the interface on <i>slot</i> and <i>port</i> .

Step	Command	Purpose
2	node(prt-eth)[slot/port]#cos tx-map <i>service class value</i> as <i>layer 2 class of service value</i>	Adds a transmit mapping table entry, which converts a <i>service class value</i> into a <i>layer 2 class of service</i> .

Example: Adding a transmit mapping table entry

The following example shows how to add a transmit mapping table entry, which converts a service class value of 4 into a layer 2 class of service value of 2 for the Ethernet port on slot 0 and port 0 of a SmartNode.

```
SN(cfg)#port ethernet 0 0
SN(prt-eth)[0/0]#cos tx-map 4 as 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 command can be used as soon as an encapsulation type is defined and the port was bound successful to an IP interface.

This procedure describes how to disable the Ethernet port on *slot* and *port*.

Mode: Configure

Step	Command	Purpose
1	node(cfg)#port ethernet <i>slot port</i>	Enters Ethernet port configuration mode for the interface on <i>slot</i> and <i>port</i>
2	node(prt-eth)[slot/port]#shutdown	Disables Ethernet port on <i>slot</i> and <i>port</i>

The **no** prefix causes to open the port 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 0 of a SmartNode 1000, 2000, or 4000 series device.

```
SN(cfg)#port ethernet 0 0
SN(prt-eth)[0/0]#shutdown
```

Checking the state of the Ethernet port on slot 0 and port 0 shows that the interface was closed.

```
SN(prt-eth)[0/1]#show port ethernet 0 1

Ethernet Configuration
-----

Port           : ethernet 0 0 1
State          : CLOSED
```

```
MAC Address      : 00:30:2B:00:1D:D4
Speed            : 10Mbps
Duplex           : Half
Encapsulation    : ip
Binding          : wan@router
Frame Format      : standard
Default Service  : 0
```

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 CLOSED for parameter state.

```
SN(prt-eth)[0/1]#show ip interface
```

```
-----
Context:          router
Name:             wan
IP Address:       172.17.100.210 255.255.255.0
MTU:              1500
ICMP router-discovery: enabled
ICMP redirect:    send only
State:            CLOSED
Binding:          ethernet 0 0 1/ethernet/ip
```


Chapter 14 **Link scheduler configuration**

Chapter contents

Introduction	152
Applying scheduling at the bottleneck	152
Using traffic classes	152
Introduction to Scheduling	153
Priority	153
Weighted fair queuing (WFQ)	153
Shaping	153
Burst tolerant shaping or wfq	154
Hierarchy	154
Quick references	155
Setting the modem rate	155
Command cross reference	156
Link scheduler configuration task list.....	156
Defining the access control list profile	157
Packet classification	157
Creating an access control list	158
Creating a service policy profile	159
Specifying the handling of traffic-classes	161
Defining fair queuing weight	161
Defining the bit-rate	162
Defining absolute priority	162
Defining the maximum queue length	162
Specifying the type-of-service (TOS) field	162
Specifying the precedence field	163
Specifying differentiated services codepoint (DSCP) marking	163
Specifying layer 2 marking	164
Defining random early detection	165
Discarding Excess Load	165
Devoting the service policy profile to an interface	166
Displaying link arbitration status	167
Displaying link scheduling profile information	167
Enable statistics gathering	167

Introduction

This chapter describes how to use and configure the SmartWare Quality of Service (QoS) features. Refer to chapter 18, “[Access control list configuration](#)” on page 203 more information on the use of access control lists.

This chapter includes the following sections:

- Quick references (see [page 155](#))
- Packet Classification (see [page 157](#))
- Assigning bandwidth to traffic classes (see [page 155](#))
- Link scheduler configuration task list (see [page 156](#))

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 SmartWare 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 SmartNode 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 SmartNode and the queuing would happen in the modem, which does distinguish between voice and data packets. To improve QoS, you can configure the SmartNode to send no more data to the Internet than the modem can carry. This keeps the modem’s queue empty and gives the SmartWare control over which packet is sent over the access link at what time.

Using traffic classes

The link scheduler 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 SmartNode has a tag attached to it on which the classification can be noted. The access control list “stage” (ACL) 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 SmartNode, 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 SmartWare: voice packets and data packets origination from or destined to the SmartNode itself are marked as “local-voice” and “local-default” respectively. Please refer to [figure 31](#) on page 153 when using the ACL to classify traffic. It illustrates the sequence of processing stages every routed packet passes. Only stages that have been installed in the data path with a “use profile...” statement in the corresponding interface configuration are present. Both an input direction ACL on the receiving interface as well as an output ACL on the transmitting interface can be used to classify a packet for special handling by the output link scheduler on the transmit interface. But as visible from the figure no ACL can be used for an input link scheduler.

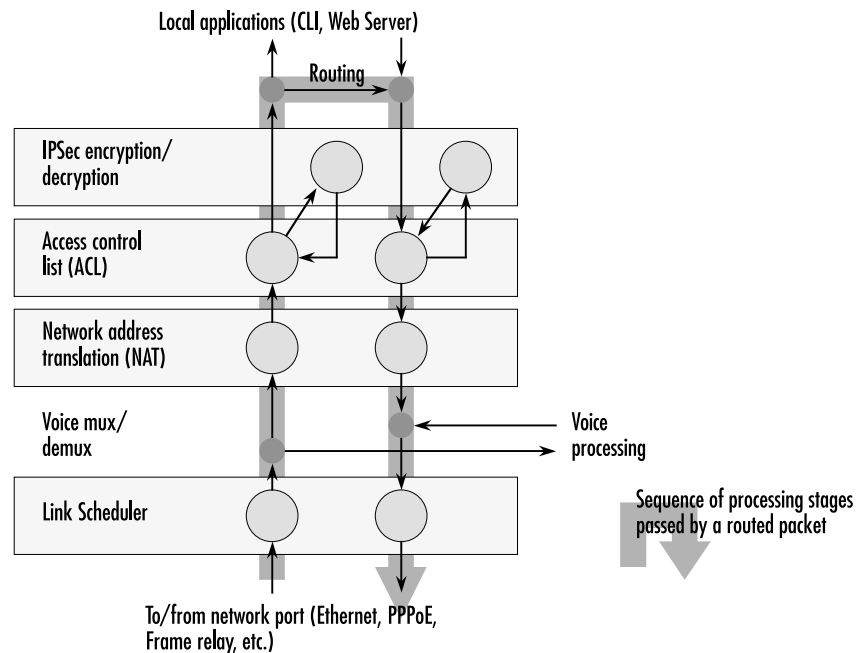


Figure 31. Packet routing in SmartWare

The QoS features in SmartWare are a combination of an access control list (used for packet classification) and a service-policy profile (used by the link arbiter to define the arbitration mode and the order in which packets of different classes are served).

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. SmartWare uses the priority scheme to make sure that voice packets generated by the SmartNode 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 SmartNode to a *Diff-Serv* network shaping might be a required operation.

Burst tolerant shaping or wfq

For weighted fair queuing and 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 SmartWare shaper assumes that the burstiness needed for sources to catch up after *collisions* is implicitly allowed. Future versions of SmartWare might allow setting the burst rate and bursting size if more control over its behavior is considered necessary.

Burst tolerance has a different effect when used with *weighted fair queuing*. Think of it as a higher initial rate when a source device starts transmitting data packets. This allows giving a higher *weight* to short data transfers. This feature is sometimes referred to as a *service curve*.

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 SmartWare with service-policy profiles used inside service-policy profiles. In [figure 32](#) 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.

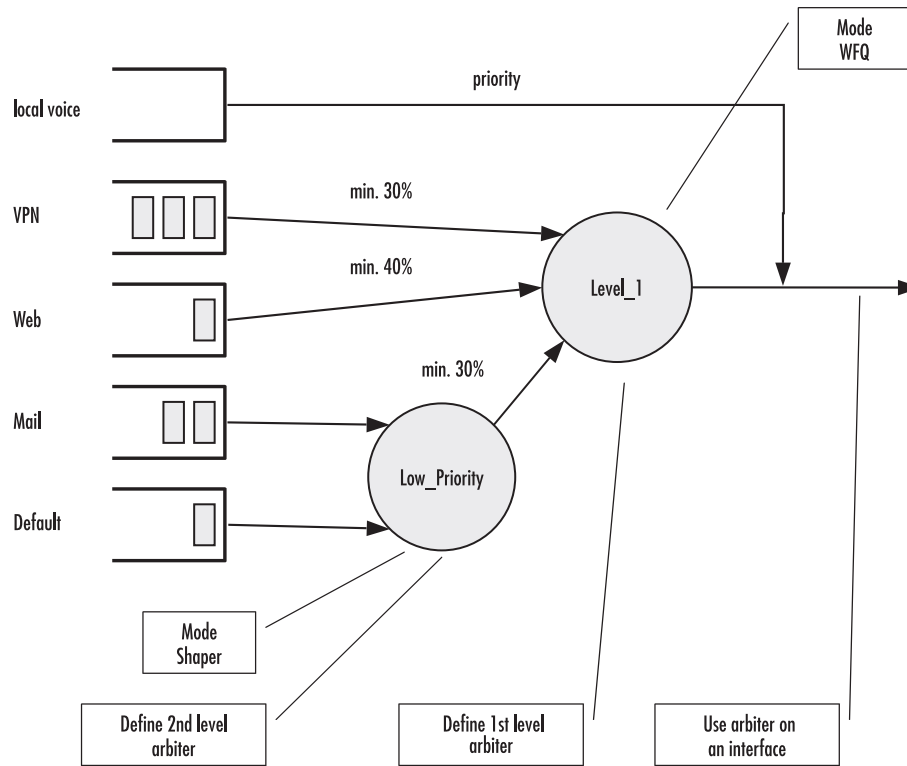


Figure 32. Example of Hierarchical Scheduling

Quick references

The following sections provide a minimal “standard” link scheduler configuration for the case where voice and data share a (DSL/cable) modem link. You will also find a command cross reference list for administrators familiar with Cisco’s IOS QoS features and having to become acquainted with SmartWare QoS configuration.

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 SmartWare link scheduler.

1. Create a minimal profile.

```
profile service-policy modem-512
  rate-limit 512 header-length 20 atm-modem
  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.
- “header-length 20” 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-modem” 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 header-length 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.

Command cross reference

Comparing SmartWare with the Cisco IOS QoS software command syntax often helps administrators to straightforwardly configure SmartNode devices. In [table 7](#) the Cisco IOS Release 12.2 QoS commands are in contrast with the respective SmartWare commands.

Table 7. Command cross reference

Action	IOS command	SmartWare command
Specifies the name of the policy map or profile to be created or modified.	policy-map policy-map-name	profile service-policy profile-name
Specifies the name of the class map or class to be created.	class-map class-map-name	source traffic-class class-name
For IOS specifies average or peak bit rate shaping. For SmartWare assigns the average bit rate to a source.	shape {average peak} cir [bc] [be]	rate bit-rate
For IOS specifies or modifies the bandwidth allocated for a class belonging to a policy map. Percent defines the percentage of available bandwidth to be assigned to the class. For SmartWare assigns the weight of the selected source (only used with wfq).	bandwidth {bandwidth-kbps percent percent}	share percent-of-bandwidth

Link scheduler configuration task list

To configure QoS features, perform the tasks described in the following sections. Depending on your requirements some of the tasks are required while other tasks are optional.

- Defining the access control list profile
- Creating a service-policy profile (see [page 159](#))
- Specifying the handling of traffic-classes (see [page 161](#))
- Devoting the service policy profile to an interface (see [page 166](#))
- Displaying link arbitration status (see [page 167](#))
- Displaying link scheduling profile information (see [page 167](#))

- Enable statistics gathering (see [page 167](#))

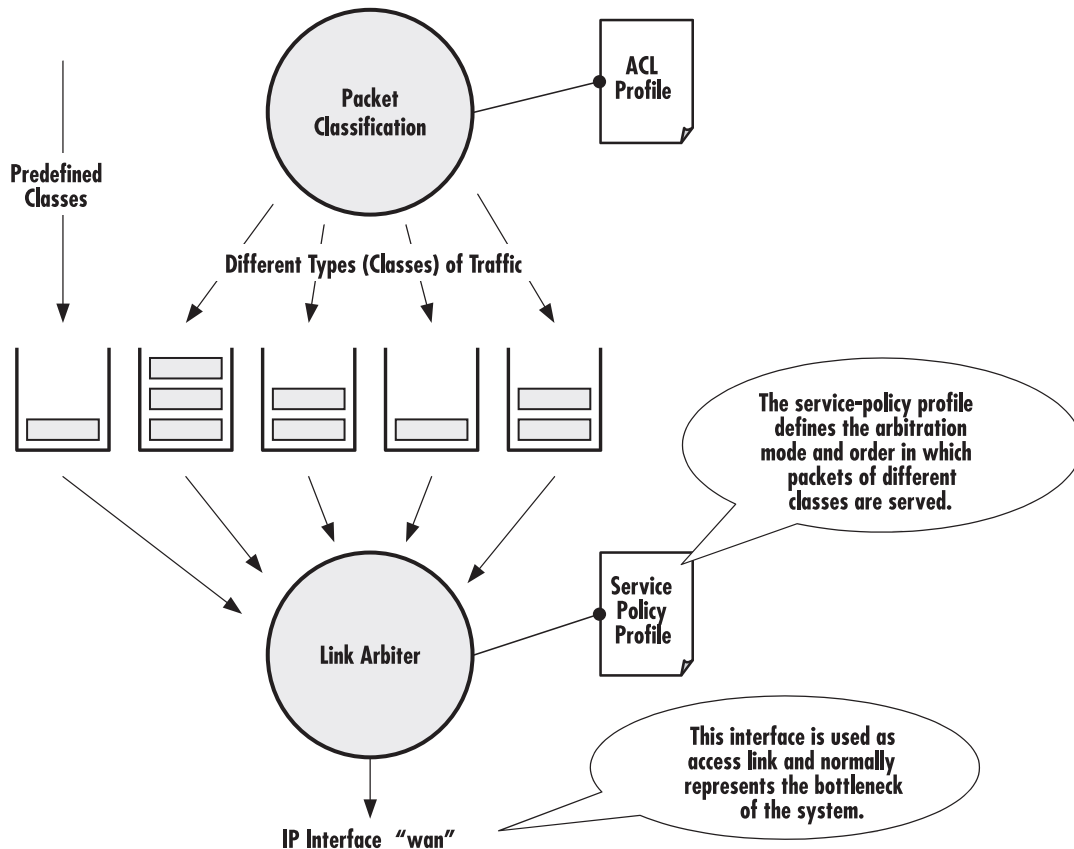


Figure 33. Elements of link scheduler configuration

Defining the access control list profile

Packet classification

The basis for providing any QoS lies in the ability of a network device to identify and group specific packets. This identification process is called *packet classification*. In SmartWare access control lists are used for packet classification.

An access control list in SmartWare consists of a series of packet descriptions like “addressed to xyz”. Those descriptions are called rules. For each packet the list of descriptions is sequentially checked and the first rule that matches decides what happens to the packet. As far as filtering is concerned the rule decides if the packet is discarded (“deny”) or passed on (“permit”). You can also add a traffic-class to the rule and if this rule is the first matching rule for a packet it is tagged with the traffic-class name.

Some types of packets you do not have to tag with ACL. Voice and data packets from of for the SmartNode itself are automatically tagged with predefined traffic-class names: Predefined internal classes for voice and other data are:

- **local-voice**—VoIP packets that originate from the SmartNode itself.

- **local-default**—All other packets that originate from the SmartNode itself.
- **default**—All traffic that has not otherwise been labeled.

Creating an access control list

The procedure to create an access control list is described in detail in chapter 18, “Access control list configuration” on page 203.

At this point a simple example is given, that shows the necessary steps to tag any outbound traffic from a Web server. The scenario is depicted in [figure 34](#). The IP address of the Web server is used as source address in the permit statement of the IP filter rule for the access control list.

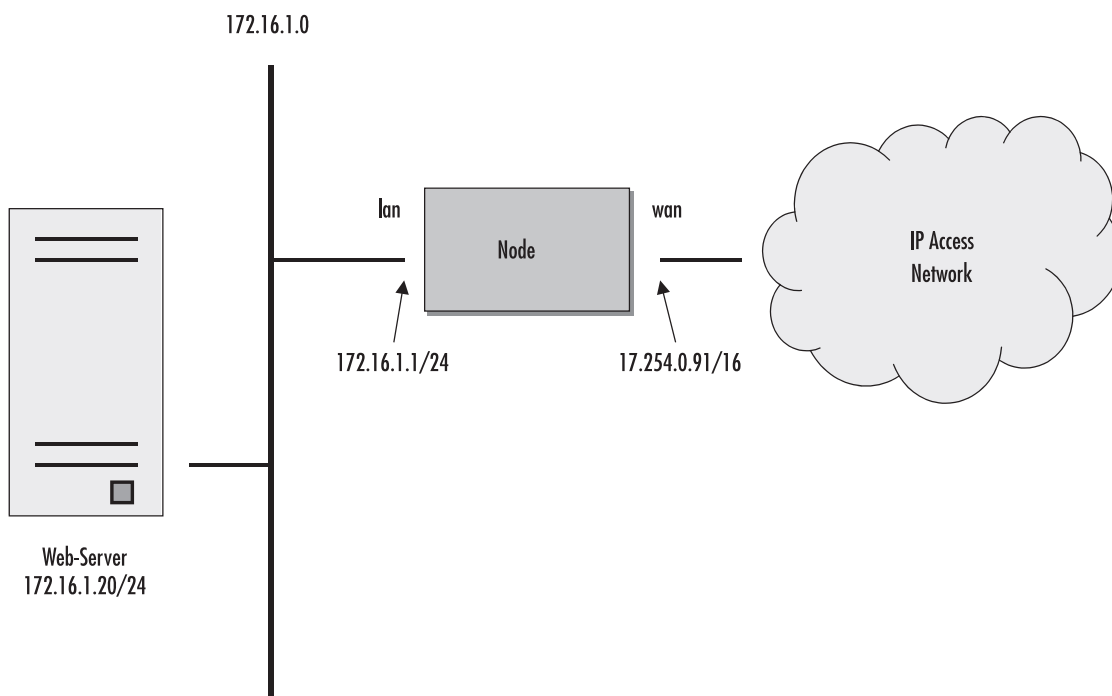


Figure 34. Scenario with Web server regarded as a single source host

A new access control list has to be created. In the example above, the traffic-class that represents outbound Web related traffic is named *Web*.

Access control list have an implicit “deny all” entry at the very end, so packets that do not match the first criteria of outbound Web related traffic will be dropped. That is why a second access control list entry—one that allows all other traffic—is necessary.

This procedure describes creating an access control list for tagging web traffic from the single source host at a certain IP address.

Mode: Configure

Step	Command	Purpose
1	node(cfg)#profile acl <i>name</i>	Creates a new access control list profile named <i>name</i>
2	node(pf-acl)[<i>name</i>] # permit ip host ip-address any traffic-class <i>class-name</i>	Creates an IP access control list entry that permits access for host at IP address <i>ip-address</i> , and specifies that packets matched by this rule belong to the traffic-class <i>class-name</i> .
3	node(pf-acl)[<i>name</i>] # permit ip any any <i>any</i>	Creates an IP access control list entry that permits IP traffic to or from all IP addresses.

Example: Defining the access control list profile

In the example below a new access control list profile named *Webserver* is created. In addition an IP access control list entry that permits access for host at IP address *172.16.1.20*, and specifies that packets matched by this rule belong to the traffic-class *Web* is added. Finally an IP access control list entry that permits IP traffic to or from all IP addresses is added to the access control list.

```
SN(cfg)#profile acl Webserver
SN(pf-acl)[Webserver~]#permit ip host 172.16.1.20 any traffic-class Web
SN(pf-acl)[Webserver~]#permit ip any any
```

After packet classification is done using access control lists, the link arbiter needs rules defining how to handle the different traffic-classes. For that purpose you create a service-policy profile. The service policy profile defines how the link arbiter has to share the available bandwidth among several traffic classes on a certain interface.

Creating a service policy profile

The service-policy profile defines how the link scheduler should handle different traffic-classes. The overall structure of the profile is as follows:

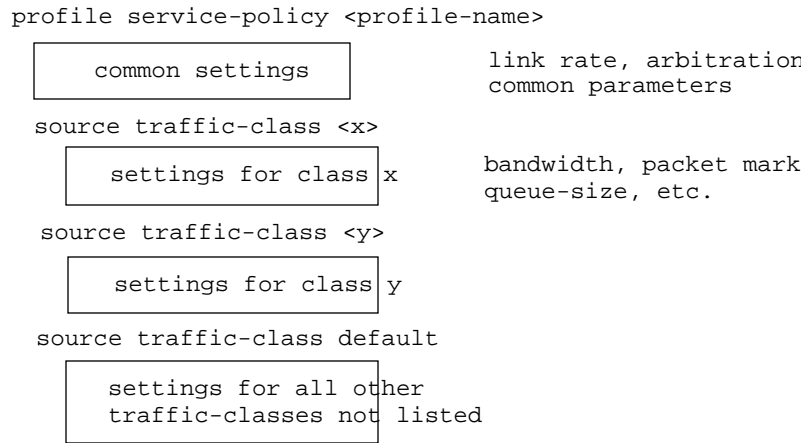


Figure 35. Structure of a Service-Policy Profile

The template shown above specifies an arbiter with three inputs which we call “sources”: x, y and “default”. The traffic-class “default” stands for all other packets that belong neither to traffic-class x nor y. There is no limit on the number of sources an arbiter can have.

Example: Creating a service policy profile

The following example shows how to create a top service-policy profile named *sample*. This profile does not include any hierarchical sub-profiles. The bandwidth of the outbound link is limited to 512 kbps therefore the interface rate-limit is set to 512. In addition weighted fair queuing (wfq) is used as arbitration scheme among the source classes.

```

profile service-policy sample
rate-limit 512
mode wfq
source traffic-class local-voice
priority
source traffic-class Web
share 30
source traffic-class local-default
share 20
source traffic-class default
queue-limit 40
share 50

```

The first line specifies the name of the link arbiter profile to configure. On the second line the global bandwidth limit is set. The value defining the bandwidth is given in kilobits per second. Each service-policy profile must have a “rate-limit” except if no scheduling is used i.e. the link scheduler is used for packet marking only (like setting the TOS byte).

How the bandwidth on an IP interface is shared among the source classes is defined on the third line. The mode command allows selecting between the weighted fair queuing and shaping arbitration mode. The default mode is wfq - the command shown above can therefore be omitted.

The following lines configure the source traffic-classes. When using weighted fair queuing (wfq) each user-specified source traffic-class needs a value specifying its share of the overall bandwidth. For this purpose the `share` command is used, which defines the relative weights of the source traffic-classes and policies.

At a some point the source traffic-class *default* must be listed. This class must be present, because it defines how packets, which do not belong to any of the traffic-classes listed in the profile are to be handled. When all listed “traffic-classes” have “priority” the handling of the remaining traffic is implicitly defined and the “default” section can be omitted. Similarly if no scheduling is used i.e. the link scheduler is used for packet marking only (e.g. setting the TOS byte) the “default” section can also be omitted.

The table below shows the basic syntax of the service-policy profile structure:

Mode: Configure

Step	Command	Purpose
1	node(cfg)# profile service-policy name	Creates a new service policy profile named name
2	node(pf-srvpl)[name]#rate-limit value	Limits global interface rate to value in kbps. Be aware, that the actual rate-limit on a given interface has to be defined for reliable operation.
3	node(pf-srvpl)[name]#mode {shaper wfq}	Sets the arbitration scheme to mode shaper or weighted fair queuing (wfq). If not specified wfq is default.
4	node(pf-srvpl)[name]#source {traffic-class policy} src-name	Enters source configuration mode for a traffic-class or a hierarchical lower level service-policy profile named src-name.
5	node (src)[src-name]...	At this point the necessary commands used to specify the handling of the traffic-class(es) have to be entered.
6	node (src)[src-name]exit	Leaves the source configuration mode (optional)
7	node(pf-srvpl)[name]#...	Repeat steps 4 to 6 for all necessary source classes or lower level service policy profiles.
8	node(pf-srvpl)[name]#exit	Leaves the service-policy profile mode

Specifying the handling of traffic-classes

Several commands are available to specify what happens to a packet of a specific traffic-class.

Defining fair queuing weight

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%.

Mode: Source

Command	Purpose
node(src)[name]#share <i>percentage</i>	Defines fair queuing weight (relative to other sources) to <i>percentage</i> for the selected class or policy <i>name</i>

Defining the bit-rate

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 overloaded the shaper will behave like a wfq arbiter. Bit-rate specification for shaper (kilobits).

Mode: Source

Command	Purpose
node(src)[name]#rate [<i>kilobits</i> remaining]	Defines the (average) bit-rate to the selected in kbps <i>kilobits</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>

Defining absolute priority

This command **priority** can only be applied to classes, but not to lower level policies. 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

Command	Purpose
node(src)[name]#priority	Defines absolute priority effectively bypassing the link arbiter for the selected class or policy <i>name</i>

Defining the maximum 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

Command	Purpose
node(src)[name]#queue-limit <i>number-of-packets</i>	Defines the maximum number of packets queued for the selected class or policy <i>name</i>

Specifying the type-of-service (TOS) field

The **set ip tos** command specifies the type-of-service (TOS) field value applied to packets of the class *name*. TOS and DSCP markings cannot be used at the same time. The **no** form of this command disables TOS marking.

The type-of-service (TOS) byte in an IP header specifies precedence (priority) and type of service (RFC791, RFC1349). The precedence field is defined by the first three bits and supports eight levels of priority. The next four bits—which are set by the `set ip tos` command—determine the type-of-service (TOS).

Table 8. TOS values and their meaning

TOS Value	SmartWare Value	Meaning
1000	8	Minimize delay.
0100	4	Maximize throughput.
0010	2	Maximizes reliability.
0001	1	Minimize monetary costs.
0000	0	All bits are cleared, normal service, "default TOS".

Historically those bits had distinct meanings but since they were never consistently applied routers will ignore them by default. Nevertheless you can configure your routers to handle specific TOS values and SmartWare allows you to inspect the TOS value in the ACL rules and to modify the TOS value with the link scheduler `set ip tos` command.

Mode: Source

Command	Purpose
<code>node(src)[name]#set ip tos value</code>	Defines the type-of-service (TOS) value applied to packets of for the selected class or policy <i>name</i> . Standard ToT values are 0, 1, 2, 4, and 8, as given in table 8 on page 163, but any number from 0 to 15 can be configured.

Specifying the precedence field

The `set ip precedence` command specifies the precedence marking applied to packets of the class name. Precedence and DSCP markings cannot be used at the same time.

The type-of-service (TOS) byte in an IP header specifies precedence (priority) and type of service (RFC791, RFC1349). The precedence field is defined by the first three bits and supports eight levels of priority. The lowest priority is assigned to 0 and the highest priority is 7.

The `no` form of this command disables precedence marking.

Mode: Source

Command	Purpose
<code>node(src)[name]#set ip precedence value</code>	Defines the precedence marking value applied to packets of for the selected class or policy <i>name</i> . The range for <i>value</i> is from 0 to 7, but only values from 0 to 5 should be used.

Specifying differentiated services codepoint (DSCP) marking

Differentiated services enhancements to the Internet protocol are intended to enable the handling of “traffic-classes” throughout the Internet. In this context the IP header TOS field is interpreted as something like a

“traffic-class” number called. With SmartWare you can inspect the DSCP value in the ACL rules and modify the DSCP value with the link scheduler `set ip dscp` command.

Note When configuring service differentiation on the SmartNode, ensure that codepoint settings are arranged with the service provider.

The command `set ip dscp` sets the DS field applied to packets of the class *name*. Additionally shaping may be needed to make the class conformant. The `no` form of this command disables packet marking.

Mode: Source

Command	Purpose
<code>node(src)[name]#set ip dscp value</code>	Defines the Differentiated Services Codepoint value applied to packets of for the selected class or policy <i>name</i> . The range for <i>value</i> is from 0 to 63.

Specifying layer 2 marking

The IEEE ratified the 802.1p standard for traffic prioritization in response to the realization that different traffic classes have different priority needs. This standard defines how network frames are tagged with user priority levels ranging from 7 (highest priority) to 0 (lowest priority). 802.1p-compliant network infrastructure devices, such as switches and routers, prioritize traffic delivery according to the user priority tag, giving higher priority frames precedence over lower priority or non-tagged frames. This means that time-critical data can receive preferential treatment over non-time-critical data.

Under 802.1p, a 4-byte Tag Control Info (TCI) field is inserted in the Layer 2 header between the Source Address and the MAC Client Type/Length field of an Ethernet Frame. Table 9 lists the tag components.

Table 9. Traffic control info (TCI) field

Tag Control Field	Description
Tagged Frame Type Interpretation	Always set to 8100h for Ethernet frames (802.3ac tag format)
3-Bit Priority Field (802.1p)	Value from 0 to 7 representing user priority levels (7 is the highest)
Canonical	Always set to 0
12-Bit 802.1Q VLAN Identifier	VLAN identification number

802.1p-compliant infrastructure devices read the 3-bit user priority field and route the frame through an internal buffer/queue mapped to the corresponding user priority level.

The command `set layer2 cos` specifies the layer 2 marking applied to packets of this class by setting the 3-bit priority field (802.1p). The `no` form of this command disables packet marking.

Please note that the Ethernet port must be configured for 802.1Q framing. Standard framing has no class-of-service field.

Mode: Source

Command	Purpose
<code>node(src)[name]#set layer2 cos value</code>	Defines the Class-Of-Service value applied to packets of for the selected class or policy <i>name</i> . The range for <i>value</i> is from 0 to 7.

Defining random early detection

The command **random-detect** is used to request random early detection (RED). When a queue carries lots of TCP transfers that last longer than simple web requests, there is a risk that TCP flow-control might be inefficient. A burst-tolerance index between 1 and 10 may optionally be specified (exponential filter weight). The **no** form of this command reverts the queue to default “tail-drop” behavior.

Mode: Source

Command	Purpose
node(src)[name]#random-detect {burst-tolerance}	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.

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.

This procedure describes defining discard excess load

Mode: Source

Command	Purpose
node(src)[name]#police <i>average-kilobits</i> burst-size <i>kilobits-ahead</i>	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.

Devoting the service policy profile to an interface

Any service policy profile needs to be bound to a certain IP interface to get activated. According the terminology of SmartWare a service policy profile is used on a certain IP interface, as shown in [figure 36](#).

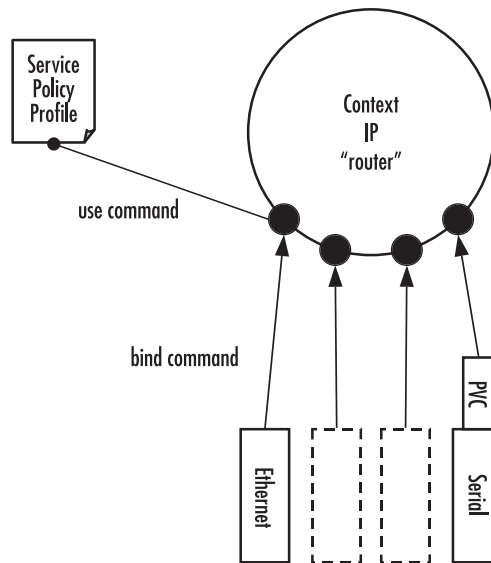


Figure 36. Using a Service Policy Profile on an IP Interface

Therefore the **use profile service-policy** command allows attaching a certain service policy profile to an IP interface that is defined within the IP context. This command has an optional argument that defines whether the service policy profile is activated in receive or transmit direction.

Providers may use input shaping to improve downlink voice jitter in the absence of voice support. The default setting “no service-policy” sets the interface to FIFO queuing.

Mode: Interface

Step	Command	Purpose
1	node(if-ip)[if-name]#use profile service-policy name {in out}	Applies the service policy profile <i>name</i> to the selected interface <i>if-name</i> . Depending on selecting the optional in or out argument the service policy profile is active on the receive or transmit direction. Be aware that service policy profiles can only be activated on the transmit direction at the moment.

Example: Devoting the service policy profile to an interface

The following example shows how to attach the service policy profile *Voice_Prio* to the IP interface wan that is defined within the IP context for outgoing traffic.

```
SN>enable
SN#configure
SN(cfg)#context ip router
```



```
SN(ctx-ip)[router]#interface wan
SN(if-ip)[wan]#use profile service-policy Voice_Prio out
```

Displaying link arbitration status

The **show service-policy** command displays link arbitration status. This command supports the optional argument **interface** that select a certain IP interface. This command is available in the operator mode.

Mode: Operator execution

Step	Command	Purpose
1	node>show service-policy {interface name}	Displays the link arbitration status

Example: Displaying link arbitration status

The following example shows how to display link arbitration status information.

```
SN>show service-policy
available queue statistics
-----
default
- packets in queue: 10
```

Displaying link scheduling profile information

The **show profile service-policy** command displays link scheduling profile information of an existing service-policy profile. This command is only available in the administrator mode.

Mode: Administrator execution

Step	Command	Purpose
1	node#show profile service-policy name	Displays link scheduling profile information of the service-policy profile <i>name</i>

Example: Displaying link scheduling profile information

The following example shows how to display link scheduling profile information of an existing service-policy profile *VoIP_Layer2_CoS*.

```
SN#show profile service-policy VoIP_Layer2_CoS
VoIP_Layer2_CoS
default (mark layer 2 cos -1)
```

Enable statistics gathering

Using the **debug queue statistics** commands enables statistic gathering of link scheduler operations.

The command has optional values (in the range of 1 to 4) that define the level of detail (see [table 10](#)).

Table 10. Values defining detail of the queuing statistics

Optional Value	Implication on Command Output
0	Statistic gathering is switched off
1	Display amount of packets <i>passed</i> (did not have to wait), <i>queued</i> (arrived earlier than rate permitted) and <i>discarded</i> (due to overflowing queue)
2	Also collects byte counts for the categories listed above
3	Also keeps track of the peek queue lengths ever reached since the last configuration change or reload
4	Adds delay time monitoring

Note The debug features offered by SmartWare require the CPU resources of your SmartNode. Therefore do not enable statistic gathering or other debug features if it is not necessary. Disable any debug feature after use with the **no** form of the command.

You can enable queue statistics for all queues of a link scheduler by placing the **debug queue statistics** command in the profile header. Queue statistics are reset whenever the configuration is changed or SmartWare is reloaded.

Mode: Source

Step	Command	Purpose
1	<code>node(src)[name]#debug queue statistics /level/</code>	Enables statistic gathering for the selected class or policy <i>name</i> . The optional argument <i>level</i> , which is in the range from 1 to 4, defines the verbosity of the command output.

Example: Enable statistics gathering for all queues of a profile

The following example shows how to enable statistic gathering for all traffic-classes

```
SN>enable
SN#configure
SN(cfg)#profile service-policy sample
SN(pf-srvpl)[sample]#debug queue statistics 4
```

Chapter 15 **Serial port configuration**

Chapter contents

Introduction	170
Serial port configuration task list	170
Disabling an interface	171
Enabling an interface	171
Configuring the serial encapsulation type	172
Configuring the hardware port protocol	173
Configuring the active clock edge	174
Enter Frame Relay mode	175
Configuring the LMI type	175
Configuring the keep-alive interval	176
Enabling fragmentation	176
Entering Frame Relay PVC configuration mode	178
Configuring the PVC encapsulation type	178
Binding the Frame Relay PVC to IP interface	179
Enabling a Frame Relay PVC	180
Disabling a Frame Relay PVC	181
Debugging Frame Relay	181
Displaying serial port information	182
Displaying Frame Relay information	183
Integrated service access	184

Introduction

This chapter provides an overview of the serial port and describes the tasks involved in its configuration through the SmartWare, it includes the following sections:

- Serial port configuration task list
- Configuration tasks
- Examples

The SmartNode 2300 device supports the V.35 and X.21 standard for synchronous serial interfaces with speeds up to 2 Mbps. The V.35 standard is recommended for speeds up to 48 kbps, although in practice it is used successfully at 4 Mbps. The X.21 standard is recommended for data interfaces transmitting at rates up to 2 Mbps and is used primarily in Europe and Japan.

The synchronous serial interface supports full-duplex operation and allows interconnection to various serial network interface cards or equipment. Refer to the getting started guide included with your SmartWare for specific information regarding the connector pinout and the selection of cables to connect with third-party equipment.

The SmartNode 2300 device supports the Frame Relay protocol on the synchronous serial interface. Frame Relay is an example of a packet-switched technology. Packet-switched networks enable end stations to dynamically share the network medium and the available bandwidth. Variable-length packets are used for more efficient and flexible transfers. These packets are then switched between the various network segments until the destination is reached. Statistical multiplexing techniques control network access in a packet-switched network. The advantage of this technique is that it provides more flexibility and more efficient use of bandwidth.

Serial port configuration task list

Perform the tasks in the following sections to configure a synchronous serial interface:

- Disabling an interface (see [page 171](#))
- Enabling an interface (see [page 171](#))
- Configuring the serial encapsulation type (see [page 172](#))
- Configuring the hardware port protocol (see [page 173](#))
- Configuring the active clock edge (see [page 174](#))
- Entering Frame Relay mode (see [page 175](#))
- Configuring the LMI type (see [page 175](#))
- Configuring the keep-alive interval (see [page 176](#))
- Enabling fragmentation (see [page 176](#))
- Entering Frame Relay PVC configuration mode (see [page 178](#))
- Configuring the PVC encapsulation type (see [page 178](#))
- Binding the Frame Relay PVC to IP interface (see [page 179](#))

- Disabling a Frame Relay PVC (see [page 181](#))
- Displaying Frame Relay information (see [page 183](#))

Disabling an interface

Before you replace a compact serial cable or attach your SmartNode to other serial equipment, use the **shutdown** command to disable the serial interfaces. This prevents anomalies and hardware faults. When you shut down an interface, it has the state *CLOSED* in the **show port serial** command display.

Note Use the **no shutdown** command to enable the serial interface after the configuration procedure.

This procedure describes how to shut down a serial interface

Mode: Administrator execution

Step	Command	Purpose
1	node(cfg)#port serial slot port	Selects the serial interface on slot and port
2	node(prt-ser)[slot/port]#shutdown	Shuts the selected interface down
3	node(prt-ser)[slot/port]#show port serial	Displays the serial interface configuration.

Example: Disabling an interface

The example shows how to disable the built-in serial interface on slot 0 and port 0 of a SmartNode. Check that *State* is set to *CLOSED* in the command output of **show port serial**.

```
SN(cfg)#port serial 0 0
SN(prt-ser)[0/0]#shutdown
SN(prt-ser)[0/0]#show port serial

Serial Interface Configuration
-----

Port          : serial 0 0 0
State         : CLOSED
Hardware Port : V.35
Transmit Edge : normal
Port Type     : DTE
CRC Type      : CRC-16
Max Frame Length: 2048
Recv Threshold : 1
Encapsulation :
```

Enabling an interface

After configuring the serial interface or connecting other serial devices to your SmartNode 2000, use the **no shutdown** command to enable the serial interfaces again. When you enable an interface, it has the state *OPENED* in the **show port serial** command display.

Note Use the **shutdown** command to disable the serial interface for any software or hardware configuration procedure.

This procedure describes how to enable a serial interface.

Mode: Administrator execution

Step	Command	Purpose
1	node(cfg)#port serial <i>slot port</i>	Selects the serial interface on slot and port
2	node(prt-ser)[slot/port]#no shutdown	Enables the interface
3	node(prt-ser)[slot/port]#show port serial	Displays the serial interface configuration.

Example: Enabling an interface

The example shows how to enable the built-in serial interface on slot 0 and port 0 of a SmartNode. Check that *State* is set to *OPENED* in the command output of **show port serial**.

```
SN(cfg)#port serial 0 0
SN(prt-ser)[0/0]#no shutdown
SN(prt-ser)[0/0]#show port serial

Serial Interface Configuration
-----

Port          : serial 0 0 0
State         : OPENED
Hardware Port : V.35
Transmit Edge : normal
Port Type     : DTE
CRC Type      : CRC-16
Max Frame Length: 2048
Recv Threshold : 1
Encapsulation :
```

Configuring the serial encapsulation type

The synchronous serial interface supports the Frame Relay serial encapsulation method.

To set the encapsulation method used by a serial interface, use the **encapsulation** interface configuration command.

This procedure describes how to set the encapsulation type of the serial interface for Frame Relay.

Mode: Administrator execution

Step	Command	Purpose
1	node(cfg)#port serial <i>slot port</i>	Selects the serial interface on <i>slot</i> and <i>port</i> .
2	node(prt-ser)[slot/port]#[no] encapsulation { framerelay ppp }	Sets the encapsulation type for the selected interface.
3	node(prt-ser)[slot/port]#show port serial	Displays the serial interface configuration.

Example: Configuring the serial encapsulation type

The following example enables Frame Relay encapsulation for the serial interface on slot 0 and port 0 of a SmartNode. Check that in the command output of **show port serial** *Encapsulation* is set to *framerelay*.

```
SN(cfg)#port serial 0 0
SN(prt-ser)[0/0]#encapsulation framerelay
SN(prt-ser)[0/0]#show port serial

Serial Interface Configuration
-----

Port          : serial 0 0 0
State         : CLOSED
Hardware Port : V.35
Transmit Edge : normal
Port Type     : DTE
CRC Type      : CRC-16
Max Frame Length: 2048
Recv Threshold : 1
Encapsulation : framerelay
```

Configuring the hardware port protocol

Before using the serial interface the hardware port protocol has to be specified. There are two command options available to select the suitable hardware port protocol:

- v35 for V.35 protocol to be used
- x21 for X.21 protocol to be used

Mode: Administrator execution

Step	Command	Purpose
1	node(cfg)#port serial <i>slot port</i>	Selects the serial interface on slot and port
2	node(prt-ser)[slot/port]#hardware-port {v35 x21}	Sets the hardware port protocol
3	node(prt-ser)[slot/port]#show port serial	Displays the serial interface configuration

Example: Configuring the hardware port protocol

The following example enables X.21 as hardware port protocol for the serial interface on slot 0 and port 0 of a SmartNode. Check that *Hardware Port* is set to *X.21* in the command output of **show port serial**.

```
SN(cfg)#port serial 0 0
SN(prt-ser)[0/0]#hardware-port x21
SN(prt-ser)[0/0]#show port serial

Serial Interface Configuration
-----

Port          : serial 0 0 0
State         : CLOSED
Hardware Port : X.21
Transmit Edge : normal
Port Type     : DTE
```

```

CRC Type           : CRC-16
Max Frame Length  : 2048
Recv Threshold    : 1
Encapsulation     : framerelay

```

Configuring the active clock edge

Depending on the system configurations—i.e. when using long cables, with certain modem types or data rates—synchronization problems may occur on the serial port. In these cases, it may be necessary to configure the clock edge on which the SmartNode transmits data.

This procedure describes how to set the active clock edge of the serial interface

Mode: Port serial

Step	Command	Purpose
1	<code>node(prt-ser)[slot/port]# transmit-data-on-edge positive</code>	Configures the serial interface to transmit on the positive edge of the clock (normal, default).
2	<code>node(prt-ser)[slot/port]# transmit-data-on-edge negative</code>	Configures the serial interface to transmit on the negative edge of the clock (inverted).

Example: Configuring the active clock edge

The following example enables to send data on the negative edge on slot 0 and port 0 of a SmartNode. Check that *Transmit Clock* is set to *inverted* in the command output of **show port serial**.

```

SN(cfg)#port serial 0 0
SN(prt-ser)[0/0]#transmit-data-on-edge negative
SN(prt-ser)[0/0]#show port serial

```

```

Serial Interface Configuration
-----

```

```

Port           : serial 0 0 0
State          : CLOSED
Hardware Port  : X.21
Transmit Edge  : inverted
Port Type     : DTE
CRC Type      : CRC-16
Max Frame Length: 2048
Recv Threshold : 1
Encapsulation  : framerelay

```


Enter Frame Relay mode

This section describes how to configure Frame Relay on the serial interface of a SmartNode, after setting the basic serial interface parameters according to the previous sections.

This procedure describes how to enter the Frame Relay configuration mode

Mode: Administrator execution

Step	Command	Purpose
1	<code>node(cfg)#port serial slot port</code>	Selects the serial interface on slot and port
2	<code>node(prt-ser)[slot/port]#framerelay</code>	Enters the Frame Relay configuration mode
3	<code>node(frm-rel)[slot/port]#</code>	Displays the Frame Relay configuration mode prompt

Example: Enter Frame Relay mode

The following example shows how to enter into the Frame Relay configuration mode for the serial interface on slot 0 and port 0 of a SmartNode.

```
SN(cfg)#port serial 0 0
SN(prt-ser)[0/0]#framerelay
SN(frm-rel)[0/0]#
```

Configuring the LMI type

For a Frame Relay network, the line protocol is the periodic exchange of local management interface (LMI) packets between the SmartNode series 2000 device and the Frame Relay provider equipment. If the SmartNode series 2000 device is attached to a public data network (PDN), the LMI type must match the type used on the public network.

You can set one of the following three types of LMIs on the SmartNode series 2000 devices:

- **ansi** for ANSI T1.617 Annex D,
- **gof** for *Group of 4*, which is the default for Cisco LMI, and
- **itu** for ITU-T Q.933 Annex A.

This procedure describes how to set the LMI type.

Mode: Frame Relay

Step	Command	Purpose
1	<code>node(frm-rel)[slot/port]#lmi-type {ansi gof itu}</code>	Sets the LMI type

Example: Configuring the LMI type

The following example sets the LMI type to ANSI T1.617 Annex D for Frame Relay over the serial interface on slot 0 and port 0.

```
SN(cfg)#port serial 0 0
SN(prt-ser)[0/0]#framerelay
SN(frm-rel)[0/0]#lmi-type ansi
```

Configuring the keep-alive interval

A keep-alive interval must be set to configure the LMI. By default, this interval is 10 seconds and, according to the LMI protocol, must be less than the corresponding interval on the switch. The keep-alive interval in seconds, which is represented by *number*, has to be in the range from 1 to 3600.

This procedure describes how to set the keep-alive interval

Mode: Frame Relay

Step	Command	Purpose
1	node(frm-rel)[slot/port]#keepalive <i>number</i>	Sets the LMI keep-alive interval

To disable keep-alives on networks that do not utilize LMI, use the **no keepalive** interface configuration command.

Example: Configuring the keep-alive interval

The following example sets the *keepalive* interval to 10 seconds for Frame Relay over the serial interface on slot 0 and port 0 of a SmartNode.

```
SN(cfg)#port serial 0 0
SN(prt-ser)[0/0]#framerelay
SN(frm-rel)[0/0]#keepalive 10
```

Enabling fragmentation

The SmartWare supports the FRF.12 interface and end-to-end fragmentation of large IP packets to reduce the delay imposed on voice packets on slow links (less than 512 kbps). As opposed to IP fragmentation (also supported by SmartWare) Frame Relay fragmentation is transparent to the IP layer. This leaves IP packets unchanged, which may be important for IP-based applications susceptible to IP fragmentation.

This procedure describes how to enable Frame Relay fragmentation

Mode: Frame Relay

Step	Command	Purpose
1	node(cfg)#port serial <i>slot port</i>	Selects the serial interface on slot and port.
2	node(prt-ser)[0/0]#framerelay	Enters Frame Relay configuration mode.
3	node(frm-rel)[0/0]#use profile service-policy <i>name out</i>	Uses the previously defined service policy profile on Frame Relay layer (and not on IP interface level) in outward direction.
4	node(frm-rel)[0/0]#fragment <i>size</i>	Defines the maximum size (in Bytes) of the Frame Relay payload (excluding Frame Relay header and trailer overhead) for all PVCs (FRF.12 interface fragmentation). See also the table below
5	node(frm-rel)[0/0]#pvc <i>dldi</i>	Enters the PVC configuration mode by assigning a DLCI number to be used on the specified virtual circuit.
6	node(pvc)[dldi]#fragment <i>size</i>	Defines the maximum size (in bytes) of the Frame Relay payload (excluding Frame Relay header and trailer overhead) for this PVC only (FRF.12 end-to-end fragmentation). See also the table below

Note For proper functioning, do not specify a scheduler mode (burst-shaper, burst-WFQ, shaper, WFQ) for the Frame Relay service policy profile. Furthermore, do not use the Frame Relay service policy profile on the IP layer, but rather on the Frame Relay layer (mode *framerelay*). Make sure voice traffic is being given priority over data (command **source class local-voice priority**).

Note FRF.12 end-to-end fragmentation and FRF.12 interface fragmentation are incompatible. Thus make sure that both ends of a Frame Relay link run the same fragmentation mode.

Note When running data and voice over a Frame Relay link, it is advisable to only configure fragmentation for the PVC that carries data traffic. This way, fragmentation protocol overhead and fragmentation processing overhead is only spent for data traffic—voice packets (whose length should be smaller than the fragmentation length) do not consume processing power and protocol overhead for fragmentation.

The purpose of end-to-end FRF.12 fragmentation is to support real-time and non-real-time data packets on lower-speed links without causing excessive delay to the real-time data. The FRF.12 Implementation Agreement defines FRF.12 fragmentation. This standard was developed to allow long data frames to be fragmented into smaller pieces (fragments) and interleaved with real-time frames. In this way, real-time and non-real-time data frames can be carried together on lower-speed links without causing excessive delay to the real-time traffic. End-to-end FRF.12 fragmentation is recommended for use on permanent virtual circuits (PVCs) that share links with other PVCs transporting voice and on PVCs transporting Voice over IP (VoIP).

The fragmentation size depends on the available bandwidth, the chosen codec, and its packet length:

- The less bandwidth available per call, the smaller the fragment size has to be configured.
- The shorter the voice packets, the smaller the fragment size can be configured.
- The smaller the fragment size, the bigger the overhead for long data packets.

The following table shows the minimum fragment size depending on the configured codec and its packet length without fragmenting the voice packets:

Codec (bytes)	Packet Period (ms)	Minimum Fragment Size
G.729	10	52
G.729	20	62
G.729	30	72
G.723	30	66
G.723	60	90
G.723	90	114
G.711	10	122
G.711	20	202
G.711	30	282

Entering Frame Relay PVC configuration mode

The permanent virtual circuit (PVC) is a virtual circuit that is permanently established. PVCs save bandwidth associated with circuit establishment and tear down in situations where certain virtual circuits must exist all the time.

The Frame Relay network provides a number of virtual circuits that form the basis for connections between stations attached to the same Frame Relay network.

The resulting set of interconnected devices forms a private Frame Relay group, which may be either fully interconnected with a complete mesh of virtual circuits, or only partially interconnected.

In either case, each virtual circuit is uniquely identified at each Frame Relay interface by a Data Link Connection Identifier (DLCI). In most circumstances, DLCIs have strictly local significance at each Frame Relay interface.

Assigning a DLCI to a specified Frame Relay sub interface on the SmartNode is done in the PVC configuration mode. The DLCI has to be in the range from 1 to 1022.

Note A maximum of eight PVCs can be defined.

This procedure describes how to enter the PVC configuration.

Mode: Frame Relay

Step	Command	Purpose
1	<code>node(frm-rel)[slot/port]#pvc dlci</code>	Enters the PVC configuration mode by assigning a DLCI number to be used on the specified sub interface

Example: Entering Frame Relay PVC configuration mode

The following example enters the configuration mode for PVC with the assigned DLCI of 1 for Frame Relay over the serial interface on slot 0 and port 0 of a SmartNode.

```
SN(cfg)#port serial 0 0
SN(prt-ser)[0/0]#framerelay
SN(frm-rel)[0/0]#pvc 1
SN(pvc)[1]#
```

Configuring the PVC encapsulation type

You must use the PVC configuration command **encapsulation rfc1490** to set the encapsulation type to comply with the Internet Engineering Task Force (IETF) standard (RFC 1490). Use this keyword when connecting to another vendor's equipment across a Frame Relay network.

This procedure describes how to set the encapsulation type to comply with RFC 1490

Mode: Frame Relay

Step	Command	Purpose
1	<code>node(frm-rel)[slot/port]#encapsulation rfc1490</code>	Sets RFC1490 PVC compliant encapsulation

Example: Configuring the PVC encapsulation type

The following example sets the encapsulation type to comply with RFC 1490 for PVC with the assigned DLCI of 1 for Frame Relay over the serial interface on slot 0 and port 0 of a SmartNode.

```
SN(cfg)#port serial 0 0
SN(prt-ser)[0/0]#framerelay
SN(frm-rel)[0/0]#pvc 1
SN(pvc)[1]#encapsulation rfc1490
```

Binding the Frame Relay PVC to IP interface

A newly created permanent virtual circuit (PVC) for Frame Relay has to be bound to an IP interface for further use. The logical IP interface has to be already defined and should be named according to the use of the serial Frame Relay PVC. If serial Frame Relay PVC shall be used as WAN access, a suitable name for the logical IP interface could be *wan* as in [figure 37](#) below.

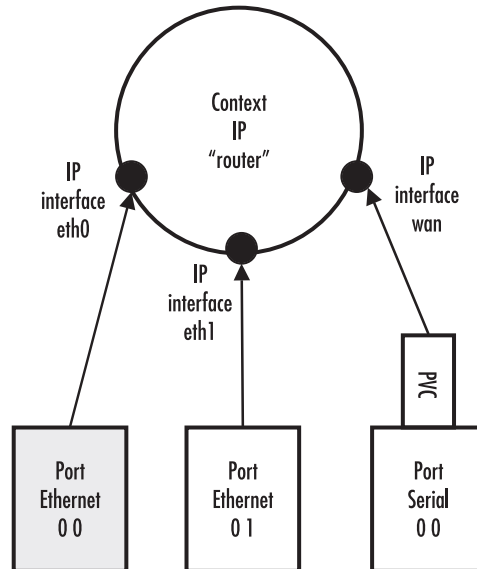


Figure 37. IP interface *wan* is bound to PVC 1 on port serial 0 0

This procedure describes how to bind the Frame Relay PVC DLCI on the serial interface to the logical IP interface *name*, which is related to the IP context router.

Mode: PVC

Step	Command	Purpose
1	<code>node(pvc)[dcli]#bind interface name router</code>	Binds Frame Relay PVC dcli to the IP interface name of IP context router

Example: Binding the Frame Relay PVC to IP interface

The following example binds the Frame Relay PVC 1 to the IP interface wan of IP context router to the serial interface on slot 0 and port 0 of a SmartNode.

```
SN(cfg)#port serial 0 0
SN(prt-ser)[0/0]#framerelay
SN(frm-rel)[0/0]#pvc 1
SN(pvc)[1]#bind interface wan router
```

Enabling a Frame Relay PVC

After binding Framrelay PVC to an ip interface it must be enabled for packet processing. This procedure activates the PVC by opening the bound ip interface.

This procedure describes how to enable Framrelay PVC for packet processing

Mode: PVC

Step	Command	Purpose
1	<code>node(pvc)[dlci]#no shutdown</code>	Enables the Frame Relay PVC

Example: Disabling a Frame Relay PVC

The following example enables Frame Relay PVC with the DLCI 1 on the serial interface on slot 0 and port 0.

```
SN(cfg)#port serial 0 0
SN(prt-ser)[0/0]#framerelay
SN(frm-rel)[0/0]#pvc 1
SN(pvc)[1]#no shutdown
```

Check the PVC 1 status using **show running-config** and verify that the entry *no shutdown* occurs in the configuration part responsible for this PVC.

```
SN(pvc)[1]#show running-config
Running configuration:
#-----#
#
#
#
pvc 1
  encapsulation rfc1490
  bind interface wan router
  no shutdown
```

Disabling a Frame Relay PVC

Frame Relay PVCs can be disabled whenever it is necessary. Be aware that disabling a specific PVC also disables the related serial interface and vice versa.

This procedure describes how to disable the Frame Relay PVC DLCI on the serial interface.

Mode: PVC

Step	Command	Purpose
1	<code>node(pvc)[dlci]#shutdown</code>	Disables the Frame Relay PVC DLCI.

Example: Disabling a Frame Relay PVC

The following example disables Frame Relay PVC 1 on the serial interface on slot 0 and port 0 of a SmartNode.

```
SN(cfg)#port serial 0 0
SN(prt-ser)[0/0]#framerelay
SN(frm-rel)[0/0]#pvc 1
SN(pvc)[1]#shutdown
```

Check the PVC 1 status by using **show running-config** and verify that the entry *shutdown* occurs in the configuration part responsible for this PVC.

```
SN(pvc)[1]#show running-config
Running configuration:
#-----#
#                                     #
# 2500                                #

pvc 1
  encapsulation rfc1490
  bind interface wan router
  shutdown
  exit
```

Debugging Frame Relay

A set of commands is available to check the status of the Framereley connections, fragmentation process and keeplive message exchange. Be aware that some monitors generate a lot of output and can seriously impact your system performance. This procedure describes how to display the Frame Relay configuration settings for the serial interface

Mode: Administrator execution

	Command	Purpose
	[no] debug framerelay	Prints the status of the different monitors (ON or OFF)
	[no] debug framerelay all	Enables/Disables all framerelay debug monitors
	[no] debug framerelay error	Enables/Disables monitor which prints only occurred errors.
	[no] debug framerelay lmi	Enables/Disables monitor which prints keepalive events and messages
	[no] debug framerelay management	Enables/Disables monitor which prints management and configuration events
	[no] debug framerelay packets	Enables/Disables monitor which prints dlcI, size and fragmentation status of every incoming and outgoing packet. Be aware that this monitor can seriously impact your system performance.

Displaying serial port information

The following example shows the commands used to display serial port configuration settings.

```
SN>enable
SN#configure
SN(cfg)#show port serial

Serial Interface Configuration
-----

Port          : serial 0 0 0
State         : OPENED
Hardware Port : X.21
Port Type     : DTE
CRC Type      : CRC-16
Max Frame Length: 2048
Recv Threshold : 1
Encapsulation : framerelay
```


Displaying Frame Relay information

Since Frame Relay configuration for the serial interface is complex and requires many commands, it is helpful to list the frame relay configuration on screen.

This procedure describes how to display the Frame Relay configuration settings for the serial interface.

Mode: Port serial

Step	Command	Purpose
1	node(prt-ser)[slot/port]#show framerelay	Displays Frame Relay information.

Example: Displaying Frame Relay information

The following example shows the commands used to display Frame Relay configuration settings.

```
SN>enable
SN#configure
SN(cfg)#show framerelay

Framerelay Configuration:
Port          LMI-Type      Keepalive      Fragmentation
-----
serial 0 0 0  ansi          10             disabled

PVC Configuration:
Port          DLCI      State      Fragment  Encaps  Binding
-----
serial 0 0 0  1        open      disabled  rfc1490  wan@router
```

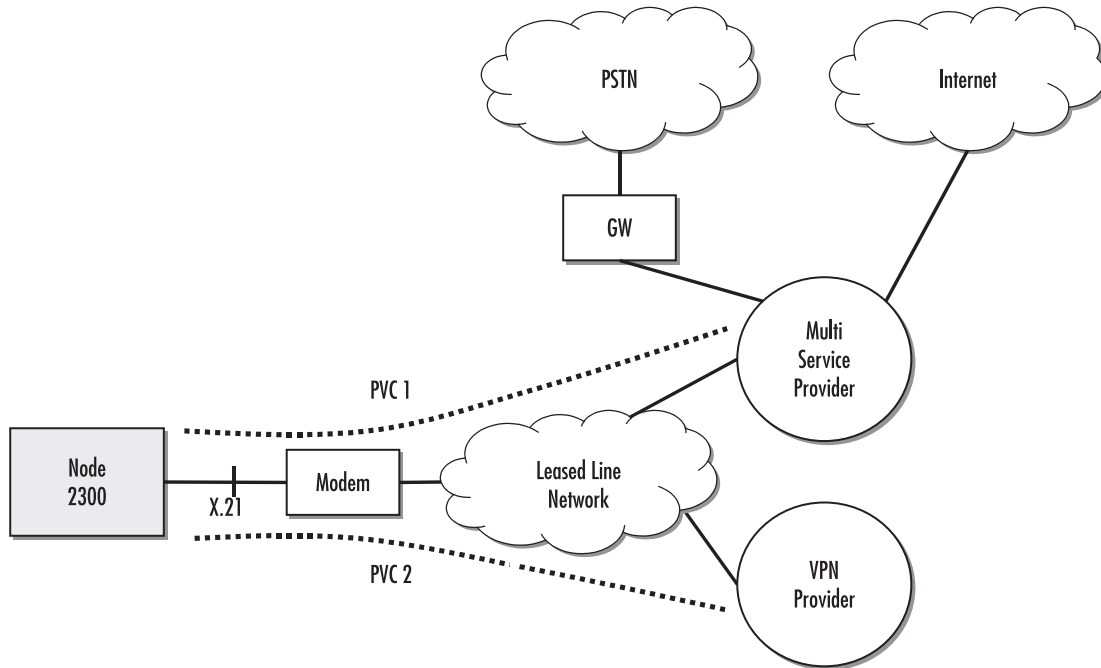


Figure 38. Typical Integrated Service Access Scenario with dedicated PVCs

Integrated service access

The example in [figure 38](#) shows a typical integrated service access scenario, where different service providers are accessed via permanent virtual circuits (PVCs) on Frame Relay over the serial interface of a SmartNode.

The multi service provider (MSP) offers both Internet access and voice services based on IP. The virtual private network (VPN) provider offers secure interconnections of local access networks (LAN) via its public wide area network based on IP. Since both providers are working independently, the SmartNode needs a configuration, which has two dedicated PVCs on Frame Relay. The first PVC, labeled as PVC 1, connects to the MSP access device. The second PVC, labeled PVC 2, connects to the VPN provider access device on the leased line network.

A SmartNode is working as a DTE and accesses the leased line network via a leased line modem connected to the serial interface. The hardware port protocol X.21 is used on the serial interface on slot 0 and port 0.

Devices accessing the MSP and VPN services are attached to the 100 Mbps Ethernet port 0/0 on the SmartNode. For that reason, an IP context with three logical IP interfaces bound to Ethernet port 0/0, PVC 1 and PVC 2 on serial port 0/0 as shown in [figure 38](#) has to be configured for the SmartNode. The IP interfaces are labeled to represent the function of their configuration. Hence Ethernet port 0/0 is named *lan*, PVC 1 is named *external* since external services are accessed via this PVC, and PVC 2 is named *internal* to indicate the private network interconnection via this PVC.

Between the leased line modem and the SmartNode, ANSI T.617 type of LMI packets have to be exchanged. In addition, the keep-alive interval has to be set to 20 seconds. To guarantee voice quality, fragmentation is enabled on the PVC which carries voice (PVC 1) and a service profile is assigned which gives priority to voices packets.

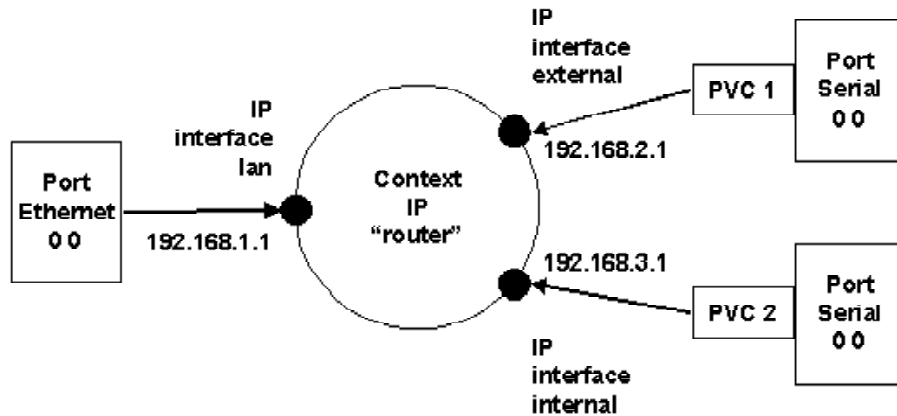


Figure 39. IP Context with logical IP interfaces bound to Ethernet port, serial port PVC 1 and PVC 2

The related IP, serial interface and Frame Relay configuration procedure is listed below. Where necessary, comments are added to the configuration for better understanding.

1. Enter the configuration mode.

```
SN>enable
SN#configure
```

2. Set up the IP interface configuration first. Be aware that not all of the necessary settings are listed below.

```
SN(cfg)#context ip router
SN(ctx-ip)[router]#interface external
SN(if-ip)[external]#interface internal
SN(if-ip)[internal]#interface lan
SN(if-ip)[lan]#exit
SN(ctx-ip)[router]#interface internal
SN(if-ip)[internal]#ipaddress 192.168.3.1 255.255.255.0
SN(if-ip)[internal]#interface external
SN(if-ip)[external]#ipaddress 192.168.2.1 255.255.255.0
SN(if-ip)[external]#interface lan
SN(if-ip)[lan]#ipaddress 192.168.1.1 255.255.255.0
```

3. Define a voice profile which gives priority to voice packets. Set the rate limit according to the bandwidth available for voice and data on PVC 1 (512kBits/s in this case).

```
SN(cfg)#profile service-policy VoicePrio
SN(pf-srvpl)[VoicePr~]#rate-limit 512
SN(pf-srvpl)[VoicePr~]#source class local-voice
SN(src)[local-v~]#priority
SN(src)[local-v~]#source class local-default
SN(src)[local-d~]#priority
SN(src)[local-d~]#source class default
```

4. Configure the serial interface settings.

```
SN(cfg)#port serial 0 0
SN(prt-ser)[0/0]#shutdown
SN(prt-ser)[0/0]#encapsulation framerelay
SN(prt-ser)[0/0]#hardware-port x21
SN(prt-ser)[0/0]#port-type dte
```

5. Configure the Frame Relay. You must thus change to the Frame Relay configuration mode. Use the service-policy profile defined above to give voice priority over data.

```
SN(prt-ser)[0/0]#framerelay
SN(frm-rel)[0/0]#lmi-type ansi
SN(frm-rel)[0/0]#keepalive 20
SN(frm-rel)[0/0]#use profile service-policy VoicePrio out
```

6. Configure the introduced PVCs. Enable fragmentation for PVC 1. The voice uses codec G.723 at a packet size of 30ms, so the minimum fragment size must be 66 Bytes. Setting the fragment size to 300 (Bytes) introduces an additional delay of at most 4.7ms ($300 * 8/512k$) but does not cause too much fragmentation overhead on large data packets.

```
SN(frm-rel)[0/0]#pvc 1
SN(pvc)[1]#encapsulation rfc1490
SN(pvc)[1]#fragment 300
SN(pvc)[1]#bind interface external router
SN(pvc)[1]#no shutdown
SN(pvc)[1]#pvc 2
SN(pvc)[2]#encapsulation rfc1490
SN(pvc)[2]#bind interface internal router
SN(pvc)[2]#no shutdown
```

7. Check that the Frame Relay settings are correct.

```
SN(frm-rel)[0/0]#show framerelay
```

Framerelay Configuration:

Port	LMI-Type	Keepalive	Fragmentation
serial 0 0 0	ansi	20	disabled

PVC Configuration:

Port	DLCI	State	Fragment	Encaps	Binding
serial 0 0 0	1	open	300	rfc1490	external@router
serial 0 0 0	2	open	disabled	rfc1490	internal@router

Chapter 16 **Basic IP routing configuration**

Chapter contents

Introduction	188
Routing tables	188
Static routing	188
Basic IP routing configuration task list	188
Configuring static IP routes	188
Deleting static IP routes	189
Displaying IP route information	190
Examples	191
Basic static IP routing example	191

Introduction

This chapter provides an overview of IP routing and describes the tasks involved in configuring static IP routing in SmartWare.

IP routing moves information across an internetwork from a source to a destination, typically passing through one or more intermediate nodes along the way. The primary difference between routing and bridging is the two different access levels of information that are used to determine how to transport packets from source to destination; routing occurs at Layer 3 (the network layer), while bridging occurs at Layer 2 (the link layer) of the OSI reference model. In addition to transporting packets through an internetwork, routing involves determining optimal paths to a destination. Routing algorithms use *metrics*, or standards of measurement, to establish these optimal paths and for initializing and maintaining routing tables that contain all route information.

Routing tables

The SmartWare routing table stores routes to:

- Directly-attached interfaces or networks
- Static IP routes
- Routes learned dynamically from the Routing Information Protocol (RIP)

In the routing table, next-hop associations specify that a destination can be reached by sending packets to a next-hop router located on an optimal path to the destination. When the SmartNode receives an incoming packet, it checks the destination address, and attempts to associate this address with a next-hop address and outgoing interface. Routing algorithms must converge rapidly — i.e. all routers must agree on optimal routes. When a network event causes routes either to go down or to become unavailable, routers distribute routing update messages that permeate networks, causing recalculation of optimal routes that are eventually agreed upon by all routers. Routing algorithms that converge slowly can cause routing loops or network outages. Many algorithms can quickly select next-best paths and adapt to changes in network topology.

Static routing

Static routing involves packet forwarding on the basis of static routes configured by the system administrator. Static routes work well in environments where network traffic is relatively predictable and where the network topology is relatively simple. In contrast, dynamic routing algorithms adjust to changing network circumstances by analyzing incoming routing update messages. RIP uses dynamic routing algorithms.

Basic IP routing configuration task list

To configure IP routes, perform the tasks described in the following sections. The tasks in the first two sections are required; the task in the remaining section is optional, but might be required for your application.

- Configuring static IP routes
- Deleting static IP routes (see [page 189](#))
- Displaying IP route information (see [page 190](#))

Configuring static IP routes

Rather than dynamically selecting the best route to a destination, you can configure one or more static routes to that destination. Once configured, a static route stays in the routing table indefinitely. When multiple static routes are configured for a single destination and the outbound interface of the current static route goes down,

a backup route is activated, thus improving network reliability. Each route is assigned a default precedence value and cost value. Modifying these values allow you to set a preference for one route over the next. If static routes are redistributed through dynamic routing protocol to neighboring devices, only the active static route to a destination is advertised.

This procedure describes how to configure one or more static IP routes to the same destination

Mode: Administrator execution

Step	Command	Purpose
1	<code>node(cfg)#context ip router</code>	Enters the IP router context
2	<code>node(ctx-ip)[router]#route network mask {address interface} [metric]</code>	Adds a static route

Where the syntax is:

- **network**—The IP address of the target network or subnet.
- **mask**—A network mask where the 1 bits indicate the network or subnet, and the 0 bits indicate the host portion of the network address provided.
- **address**—The IP address of a next-hop router that can access the target network or subnet.
- **interface**—The name of the outgoing interface to use for the target network or subnet.
- **metric**—This is an optional parameter. Specifies the desirability of the route when compared against other routes. The range is 0 through 15, where 0 is the preferred route. If no metric is specified, the static route is assumed to have a metric of 0.

Note To configure a default static IP route, use 0.0.0.0 for the network number and mask. A valid next-hop address or interface is required.

Example: Adding a static IP route

In the following example, packets for network 20.0.0.0/24 will be routed to the device at 172.17.100.2. The Ethernet port 0 1 has the address 172.17.100.1/24 and is bound to the interface *wan*.

```
SN>enable
SN#configure
SN(cfg)#context ip router
SN(ctx-ip)[router]#route 20.0.0.0 255.255.255.0 172.17.100.2
```

The route is added to the routing database with the default metric 0. The router will forward packets to the 20.0.0.0 network via the interface *wan* to the router on 172.17.100.2.

Deleting static IP routes

The **no** form of the **route** command deletes a static IP route from the routing table.

This procedure describes how to delete one or more static IP routes from the routing table

Mode: Administrator execution

Step	Command	Purpose
1	node(cfg)#context ip router	Enters the IP router context
2	node(ctx-ip)[router]#no route network mask {address interface}	Deletes a static route

Example: Deleting a static IP route

In the following example, the route for packets to network 20.0.0.0/24, which are routed to device with IP address 172.17.100.2, shall be deleted.

```
SN>enable
SN#configure
SN(cfg)#context ip router
SN(ctx-ip)[router]#no route 20.0.0.0 255.255.255.0 172.17.100.2
```

Displaying IP route information

This procedure describes how to display static IP routes

Mode: Operator or administrator execution

Step	Command	Purpose
1	node>show ip route	Displays IP route information

This command displays the destination address, next-hop interface, protocol (local, static, RIP, or ICMP), metric, flags (*U*–up, *H*–host, *G*–Gateway, *L*–local, *D*–default), and amount of use for each route in the routing table. If there are multiple routes to the same destination, the preferred route is indicated by an asterisk (*).

Example: Displaying IP route

In the following example, IP route information is displayed.

```
SN>show ip route
Routes of IP context 'router':
Status codes: * valid, U up, H host, G Gateway, L local, D default
  Destination      Nexthop      Protocol  Metric  Flags    Used
-----
* 127.0.0.1/32          local         0      LHG      n/a
* 172.16.40.77/32       local         0      LHG      n/a
* 172.17.100.210/32     local         0      LHG      n/a
* 172.17.100.0/24       wan           1      UL       0
* 20.0.0.0/24           172.17.100.2 static        0      U        0
* 172.16.0.0/16         lan           1      UL       6
```


Examples

Basic static IP routing example

Figure 40 shows an Internetwork consisting of three routers, a SmartNode 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 SmartNode shall be configured for the following IP routing scenario:

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*.

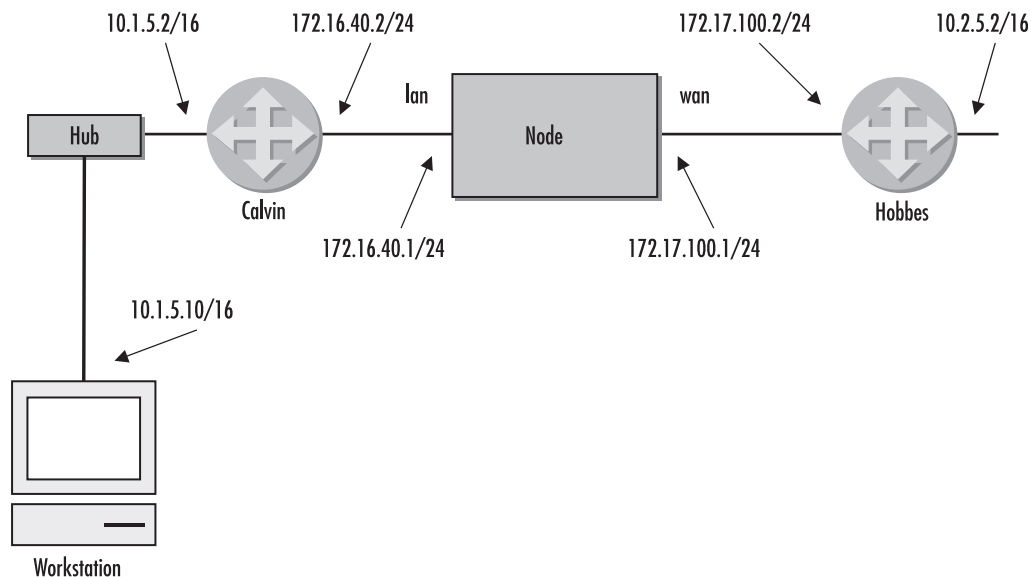


Figure 40. Internetwork with three routers and four networks

The necessary routing-table entries for the scenario described are listed below.

```
SN>enable
SN#configure
SN(cfg)#context ip router
SN(ctx-ip)[router]# route 10.1.5.10 255.255.255.255 172.16.40.2
SN(ctx-ip)[router]# route 10.2.0.0 255.255.0.0 172.17.100.2
SN>show ip route
Routes of IP context 'router':
Status codes: * valid, U up, H host, G Gateway, L local, D default
  Destination      Nexthop          Protocol  Metric  Flags    Used
-----
* 127.0.0.1/32    local            local     0       LHG      n/a
* 172.16.40.1/24  local            local     0       LHG      n/a
* 172.17.100.1/24 local            local     0       LHG      n/a
* 172.17.100.0/24 wan              local     1       UL       0
* 172.16.40.0/16 lan              local     1       UL       0
* 10.1.5.10/32    172.16.40.2     static    0       U        0
* 10.2.0.0/16    172.17.100.2   static    0       U        0
```


Chapter 17 **RIP configuration**

Chapter contents

Introduction	194
Routing protocol	194
RIP configuration task list	195
Enabling send RIP	195
Enabling an interface to receive RIP	196
Specifying the send RIP version	196
Specifying the receive RIP version	197
Enabling RIP learning	197
Enabling an interface to receive RIP	198
Enabling RIP announcing	198
Enabling RIP auto summarization	199
Specifying the default route metric	199
Enabling RIP split-horizon processing	200
Enabling the poison reverse algorithm	200
Enabling holding down aged routes	201
Displaying RIP configuration of an IP interface	201
Displaying global RIP information	202

Introduction

This chapter provides an overview of the Routing Information Protocol (RIP) and describes the tasks involved in configuring RIP features within SmartWare, it includes the following sections:

- Routing protocol
- RIP configuration task list (see [page 195](#))

RIP is a relatively old but still commonly used interior gateway protocol created for use in small, homogeneous networks. It is a classical distance-vector routing protocol. RIP is documented in RFC 1058.

RIP uses broadcast User Datagram Protocol (UDP) data packets to exchange routing information. The SmartWare software sends routing information updates every 30 seconds, which is termed *advertising*. If a router does not receive an update from another router for 180 seconds or more, it marks the routes served by the non-updating router as being unusable. If there is still no update after 240 seconds, the router removes all routing table entries for the non-updating router.

The metric that RIP uses to rate the value of different routes is the *hop count*. The hop count is the number of routers that can be traversed in a route. A directly connected network has a metric of zero; an unreachable network has a metric of 16. This small range of metrics makes RIP an unsuitable routing protocol for large networks.

A SmartNode that is running RIP can receive a default network via an update from another router that is running RIP, or the router can source (generate) the default network itself with RIP. In both cases, the default network is advertised through RIP to other RIP neighbors.

SmartWare software will send and receive RIP information from the specified interface if the following conditions are met:

- The **rip supply** flag for a specific interface is enabled
- The **rip listen** flag for a specific interface is enabled

The default route is learned via a static route and then redistributed into RIP.

RIP sends updates to the specified interfaces. If an interface is not specified, it will not be advertised in any RIP update.

Routing protocol

Routers exchange information about the most effective path for packet transfer between various end points. There are a number of different protocols, which have been defined to facilitate the exchange of this information.

Routing Information Protocol (RIP) 1 is the most widely used routing protocol on IP networks. All gateways and routers that support RIP 1 periodically broadcast routing information packets. These RIP 1 packets contain information concerning the networks that the routers and gateways can reach as well as the number of routers/gateways that a packet must travel through to reach the receiving address.

RIP 2 is an enhancement of RIP 1 which allows IP subnet information to be shared among routers, and provides for authentication of routing updates. When this protocol is chosen, the router will use the multicast address 224.0.0.9 to send and/or receive RIP 2 packets for this network interface. As with RIP 1, the router's routing table will be periodically updated with information received in these packets.

RIP 2 is more useful in a variety of environments and allows the use of variable subnet masks on your network. It is also necessary for implementation of *classless* addressing as accomplished with CIDR (classless inter-domain routing).

It is recommended that RIP 2 be used on any segment where all routers can use the same IP routing protocol. If one or more routers on a segment must use RIP 1, then all other routers on that segment should also be set to use RIP 1.

RIP configuration task list

To configure RIP, perform the tasks described in the following sections. The tasks in the first two sections are required; the tasks in the remaining sections are optional. Most of the RIP commands have the character of a flag, which is either enabled or disabled.

- Enabling send RIP
- Enabling an interface to receive RIP (see [page 196](#))
- Specifying the send RIP version (see [page 196](#))
- Specifying the receive RIP version (see [page 197](#))
- Enabling RIP learning (see [page 197](#))
- Enabling an interface to receive RIP (see [page 198](#))
- Enabling RIP announcing (see [page 198](#))
- Enabling RIP auto summarization (see [page 199](#))
- Specifying the default route metric (see [page 199](#))
- Enabling RIP split-horizon processing (see [page 200](#))
- Enabling the poison reverse algorithm (see [page 200](#))
- Enabling holding down aged routes (see [page 201](#))
- Displaying RIP Configuration of an IP interface (see [page 201](#))
- Displaying global RIP information (see [page 202](#))

Enabling send RIP

By default an interface does not send any routing information. This procedure describes how to enable sending RIP packets on interface

Mode: Interface

Step	Command	Purpose
1	<code>node(if-ip)[name]#rip supply</code>	Enables send RIP on interface <i>name</i>

Example: Enabling send RIP

The following example shows how to enable send RIP on IP interface *wan* on a SmartNode.

```
SN(cfg)#context ip router
SN(ctx-ip)[router]#interface wan
SN(if-ip)[wan]#rip supply
```

Enabling an interface to receive RIP

By default an interface does not listen to routing information. This procedure describes how to enable interface to receive RIP information

Mode: Interface

Step	Command	Purpose
1	node(if-ip)[name]#rip receive	Enables receive RIP on interface <i>name</i>

Example: Enabling receive RIP

The following example shows how to enable receive RIP on IP interface *wan* on a SmartNode.

```
SN(cfg)#context ip router
SN(ctx-ip)[router]#interface wan
SN(if-ip)[wan]#rip receive
```

Specifying the send RIP version

By default, SmartWare application software sends RIP 1 compatible packets. The SmartWare application software allows sending RIP version 1, version 1 compatible or version 2 packets. Alternatively, you can explicitly configure the RIP version to be sent with the last command argument as following:

- 1—RIPv1
- 1compatible—RIPv1 compatible
- 2—RIPv2

This procedure describes how to select the sending RIP version on interface

Mode: Interface

Step	Command	Purpose
1	node(if-ip)[name]# rip send version {1 1compatible 2}	Selects send RIP version for interface <i>name</i>

Example: Specifying the send RIP

The following example shows how to select send RIP version *1compatible* on IP interface *wan* on a SmartNode.

```
SN(cfg)#context ip router
SN(ctx-ip)[router]#interface wan
SN(if-ip)[wan]#rip send version 1compatible
```

Specifying the receive RIP version

By default, SmartWare application software receives RIP version 1 and version 2 packets. SmartWare application software allows receiving RIP version 1, version 2 or both version 1 and version 2 packets. Alternatively, you can explicitly configure the RIP version to be received with the last command argument as following:

- **1**—to receive RIP version 1 packets
- **1or2**—to receive RIP version 1 and version 2 packets
- **2**—to receive RIP version 2 packets

This procedure describes how to set receiving RIP version on an interface

Mode: Interface

Step	Command	Purpose
1	<code>node(if-ip)[name]# rip receive version {1 1or2 2}</code>	Selects receive RIP version for interface <i>name</i>

Example: Specifying the receive RIP

The following example shows how to select receive RIP version *1or2* on IP interface *wan* on a SmartNode.

```
SN(cfg)#context ip router
SN(ctx-ip)[router]#interface wan
SN(if-ip)[wan]#rip receive version 1or2
```

Enabling RIP learning

A new route is added to the local routing table, if the routing update contains a route to a destination that does not already exist. If the update describes a route whose destination is already in the local table, the new route is used only if it has a lower cost. The cost of a route is determined by adding the cost of reaching the gateway that sent the update to the metric contained in the RIP update packet. If the total metric is less than the metric of the current route, the new route is used. SmartWare offers two RIP learning mechanisms, which are represented by a specific argument of the command **rip learn**:

- **host**—for RIP learn host and
- **default**—for RIP learn default

See the following sections on how to configure those two RIP learning mechanisms.

This procedure describes how to enable accepting of IP host and default routes received on an interface for RIP learning

Mode: Interface

Step	Command	Purpose
1	<code>node(if-ip)[name]# rip learn host</code>	Enables accepting of IP host routes received on interface <i>name</i>
2	<code>node(if-ip)[name]#rip learn default</code>	Enables learning using a default route advertised by an RIP neighbor on interface <i>name</i>

Example: Enabling RIP learn host and default

The following example shows how to enable RIP learn host and default on IP interface *wan* on a SmartNode.

```
SN(cfg)#context ip router
SN(ctx-ip)[router]#interface wan
SN(if-ip)[wan]#rip learn host
SN(if-ip)[wan]#rip learn default
```

Enabling an interface to receive RIP

This procedure describes how to enable receive RIP on an IP interface

Mode: Interface

Step	Command	Purpose
1	<code>node(if-ip)[name]#rip listen</code>	Enables receive RIP on IP interface <i>name</i>

Example: Enables an interface to receive RIP

The following example shows how to enable receive RIP for IP interface *lan* on a SmartNode 1000, 2000 or 4000 series device.

```
SN(cfg)#context ip router
SN(ctx-ip)[router]#interface lan
SN(if-ip)[lan]#rip listen
```

Enabling RIP announcing

The RIP protocol supports announcing features, which are used to proclaim specific routing information to other elements, e.g. routers or SmartNodes in a network. The RIP announcing command is used for this purpose and offers options for

- **default**—for RIP default routes,
- **host**—for IP host routes,
- **self-as-default**—for self as RIP default routes and
- **static**—for static IP routes.

Depending on the RIP announcing method the last option for the command in 3 must be explicitly selected. It is possible to have more than one RIP announcing method enabled concurrently.

This procedure describes how to enable RIP announcing on an interface

Mode: Interface

Step	Command	Purpose
1	<code>node(if-ip)[name]#rip announce {default host self-as-default static}</code>	Selects the RIP announcing method on interface <i>name</i>

Example: Enabling RIP announcing

The following example shows how to enable the RIP default routes and IP host routes RIP announcing method on IP interface *wan* on a SmartNode.

```
SN(cfg)#context ip router
SN(ctx-ip)[router]#interface wan
SN(if-ip)[wan]#rip announce default
SN(if-ip)[wan]#rip announce host
```

Enabling RIP auto summarization

Summarizing routes in RIP Version 2 improves scalability and efficiency in large networks.

Auto-summarization attempts to automatically summarize groups of adjacent routes into single entries, the goal being to reduce the total number of entries in the RIP routing table, reducing the size of the table and allowing the router to handle more routes.

RIP auto-summarization (automatic network number summarization) is disabled by default. With auto-summarization, the SmartNode summarizes sub prefixes to the Class A, Class B, and Class C network boundary when class network boundaries are crossed.

This procedure describes how to enable RIP auto-summarization on an interface

Mode: Interface

Step	Command	Purpose
1	<code>node(if-ip)[name]#rip auto-summary</code>	Enables RIP auto-summarization on interface <i>name</i>

Example: Enabling RIP auto-summarization

The following example shows how to enable auto-summarization on IP interface *wan* on a SmartNode.

```
SN(cfg)#context ip router
SN(ctx-ip)[router]#interface wan
SN(if-ip)[wan]#rip auto-summary
```

Specifying the default route metric

RIP uses a single routing metric (hop count) to measure the distance between the source and a destination network. Each hop in a path from source to destination is assigned a hop-count value, which is typically 1. When a SmartNode receives a routing update that contains a new or changed destination-network entry, the SmartNode adds one to the metric value indicated in the update and enters the network in the routing table. The IP address of the sender is used as the next hop.

RIP prevents routing loops from continuing indefinitely by implementing a limit on the number of hops allowed in a path from the source to a destination. The maximum number of hops in a path is 15. If a SmartNode receives a routing update that contains a new or changed entry, and if increasing the metric value by one causes the metric to be infinity (i.e. 16), the network destination is considered unreachable.

Because metrics cannot be directly compared, you must specify the default metric in order to designate the cost of the redistributed route used in RIP updates. All routes that are redistributed will use the default metric.

Setting the default route metric, which is a number, indicating the distance to the destination network element, e.g. another router or SmartNode in a network, is possible with the **rip default-route-value** command. The value is between 1 and 15 for a valid route, or 16 for an unreachable route.

This procedure describes how to set the routing metric on an interface

Mode: Interface

Step	Command	Purpose
1	node(if-ip)[name]#rip default-route-value value	Sets the routing metric to <i>value</i> indicating the distance to the destination on interface <i>name</i>

Example: Specifying the default route metric

The following example shows how to set the routing metric to 4 on IP interface wan on a SmartNode.

```
SN(cfg)#context ip router
SN(ctx-ip)[router]#interface wan
SN(if-ip)[wan]#rip default-route-value 4
```

Enabling RIP split-horizon processing

Normally, routers that are connected to broadcast-type IP networks and that use distance-vector routing protocols employ the *split horizon* mechanism to reduce the possibility of routing loops. Split horizon blocks information about routes from being advertised by a router out of any interface from which that information originated. This behavior usually optimizes communications among multiple routers, particularly when links are broken. However, with non-broadcast networks (such as Frame Relay), situations can arise for which this behavior is less than ideal. For these situations, you might want to disable split horizon for RIP.

This procedure describes how to enable split horizon on an interface

Mode: Interface

Step	Command	Purpose
1	node(if-ip)[name]#rip split-horizon	Enables RIP split-horizon processing on interface <i>name</i>

Example: Enabling RIP split-horizon processing

The following example shows how to enable split horizon on IP interface wan on a SmartNode.

```
SN(cfg)#context ip router
SN(ctx-ip)[router]#interface wan
SN(if-ip)[wan]#rip split-horizon
```

Enabling the poison reverse algorithm

Normally, RIP uses a technique called split horizon to avoid routing loops and allow smaller update packets. This technique specifies that when the router sends a RIP update out a particular network interface, it should never include routing information acquired over that same interface.

There is a variation of the split horizon technique called *poison reverse* which specifies that all routes should be included in an update out a particular interface, but that the metric should be set to infinity for those routes

acquired over that interface. Poison reverse updates are then sent to remove the route and place it in hold-down. One drawback is that routing update packet sizes will be increased when using poison reverse.

This procedure describes how to enable the poison reverse algorithm on an interface

Mode: Interface

Step	Command	Purpose
1	node(if-ip)[name]#rip poison-reverse	Enables the poison reverse algorithm on interface <i>name</i>

Example: Enabling the poison reverse algorithm

The following example shows how to enable the poison reverse algorithm on IP interface *wan* on a Smart-Node.

```
SN(cfg)#context ip router
SN(ctx-ip)[router]#interface wan
SN(if-ip)[wan]#rip poison-reverse
```

Enabling holding down aged routes

Holding down or locking aged routes learned from RIP packets on the specified interface helps, if an aged route cannot be refreshed to a non-aged status but must be deleted and then relearned. Enabling this function enhances the stability of the RIP topology in the presence of transients.

This procedure describes how to enable holding down of aged routes on an interface

Mode: Interface

Step	Command	Purpose
1	node(if-ip)[name]#rip route-holddown	Enables holding down aged routes on interface <i>name</i>

Example: Enabling holding down aged routes

The following example shows how to enable holding down of aged routes on IP interface *wan* on a Smart-Node.

```
SN(cfg)#context ip router
SN(ctx-ip)[router]#interface wan
SN(if-ip)[wan]#rip route-holddown
```

Displaying RIP configuration of an IP interface

Displaying the RIP configuration of an IP interface is useful to list the settings. This procedure describes how to display the RIP configuration of an interface

Mode: Interface

Step	Command	Purpose
1	node(if-ip)[name]#show rip interface ifname	Displays the RIP binding of an IP interface on <i>name</i>

Example: Displaying RIP configuration of an IP interface

The following example shows how to display the RIP configuration of IP interface *wan* of a SmartNode.

```
SN(cfg)#context ip router
SN(ctx-ip)[router]#interface wan
SN(if-ip)[wan]#show rip interface wan
Interface wan (IP context router):
-----
                listen: disabled
                supply: enabled
                send version: 1compatible
receive version: 1or2
                learn host: disabled
                learn default: disabled
                announce host: disabled
                announce static: disabled
                announce default: disabled
announce self-as-default: disabled
                route-holddown: enabled
                poison-reverse: disabled
                auto-summary: disabled
                split-horizon: disabled
                default-route-value: 0
-----
```

Displaying global RIP information

SmartWare also support displaying global RIP information for the IP router context. This procedure describes how to display the global RIP information

Mode: Configure

Step	Command	Purpose
1	node(cfg)#show rip	Displays the RIP information

Example: Displaying global RIP information

The following example shows how to display the global RIP information on a SmartNode.

```
SN(cfg)#show rip
RIP information:
rip enabled
```

Chapter 18 **Access control list configuration**

Chapter contents

Introduction	204
About access control lists	204
What access lists do	204
Why you should configure access lists	204
When to configure access lists	205
Features of access control lists	205
Access control list configuration task list	206
Mapping out the goals of the access control list	206
Creating an access control list profile and enter configuration mode	207
Adding a filter rule to the current access control list profile	207
Adding an ICMP filter rule to the current access control list profile	209
Adding a TCP, UDP or SCTP filter rule to the current access control list profile	211
Binding and unbinding an access control list profile to an IP interface	213
Displaying an access control list profile	214
Debugging an access control list profile	214
Examples	216
Denying a specific subnet	216

Introduction

This chapter provides an overview of IP Access Control Lists and describes the tasks involved in configuring them through SmartWare.

This chapter includes the following sections:

- About access control lists
- Access control list configuration task list (see [page 206](#))
- Examples (see [page 216](#))

About access control lists

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

Access lists filter network traffic by controlling whether routed packets are forwarded, dropped or blocked at the router's interfaces. Your router examines each packet to determine whether to forward or drop the packet, based on the criteria you specified within the access lists.

Access list criteria could be the source address of the traffic, the destination address of the traffic, the upper-layer protocol, or other information.

Note Sophisticated users can sometimes successfully evade or fool basic access lists because no authentication is required.

Why you should configure access lists

There are many reasons to configure access lists. For example, you can use access lists to restrict contents of routing updates, or to provide traffic flow control. But one of the most important reasons to configure access lists is to provide security for your network, and this is the reason explored in this chapter.

You should use access lists to provide a basic level of security for accessing your network. If you do not configure access lists on your router, all packets passing through the router could be allowed onto all parts of your network.

For example, access lists can allow one host to access a part of your network, and prevent another host from accessing the same area. In [figure 41](#) host A is allowed to access the Human Resources network and host B is prevented from accessing the Human Resources network.

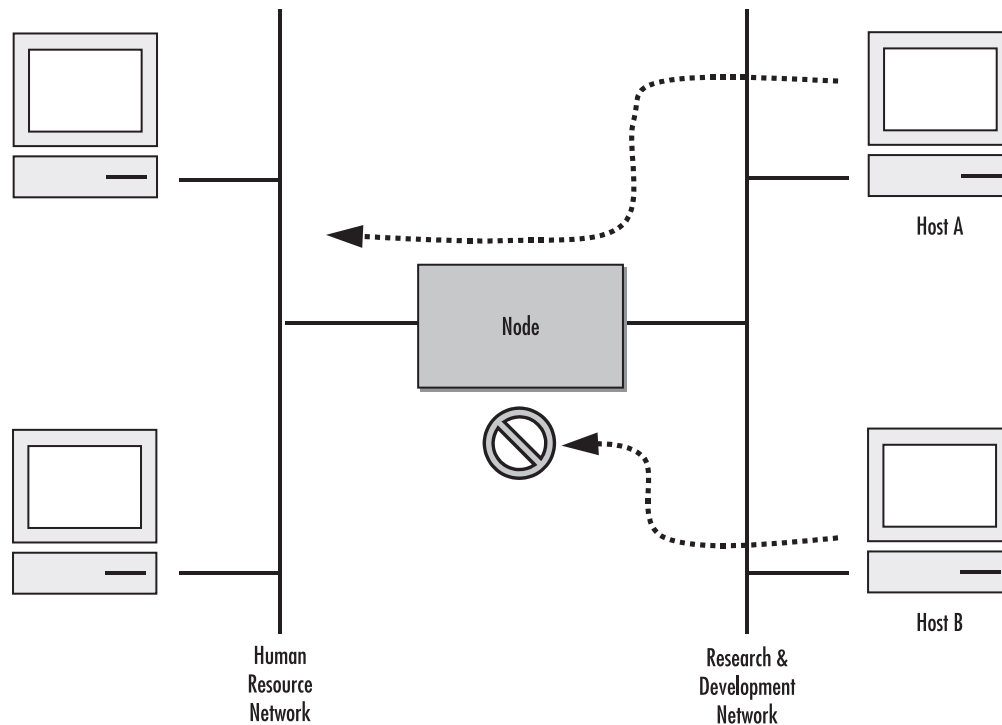


Figure 41. Using traffic filters to prevent traffic from being routed to a network

You can also use access lists to decide which types of traffic are forwarded or blocked at the router interfaces. For example, you can permit e-mail traffic to be routed but at the same time block all Telnet traffic.

When to configure access lists

Access lists 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 access lists 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 access lists, you should configure access lists 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 access lists for each network protocol configured on the router interfaces. You can configure access lists 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 access control lists:

- A list may contain multiple entries. The order access of control list entries is significant. Each entry is processed in the order it appears in the configuration file. As soon as an entry matches, the corresponding action is taken and no further processing takes place.

- All access control lists have an implicit *deny ip any any* at the end. A packet that does not match the criteria of the first statement is subjected to the criteria of the second statement and so on until the end of the access control list is reached, at which point the packet is dropped.
- Filter types include IP, Internet Control Message Protocol (ICMP), Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and Stream Control Transmission Protocol (SCTP).
- An empty access control list is treated as an implicit *deny ip any any* list.

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.

To delete an entire access control list, enter configuration mode and use the **no** form of the **profile acl** command, naming the access list to be deleted, e.g. `no profile acl name`. To unbind an access list from the interface to which it was applied, enter the IP interface mode and use the **no** form of the access control list command.

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 access control list
- Creating an access control list profile and enter configuration mode (see [page 207](#))
- Adding a filter rule to the current access control list profile (see [page 207](#))
- Adding an ICMP filter rule to the current access control list profile (see [page 209](#))
- Adding a TCP, UDP or SCTP filter rule to the current access control list profile (see [page 211](#))
- Binding and unbinding an access control list profile to an IP interface (see [page 213](#))
- Displaying an access control list profile (see [page 214](#))
- Debugging an access control list profile (see [page 214](#))

Mapping out the goals of the access control list

To create an access control list you must:

- Specify the protocol to be filtered
- Assign a unique name to the access list
- Define packet-filtering criteria

A single access control list can have multiple filtering criteria statements.

Before you begin to enter the commands that create and configure the IP access control list, be sure that you are clear about what you want to achieve with the list. Consider whether it is better to deny specific accesses and permit all others or to permit specific accesses and deny all others.

Note Since a single access control list can have multiple filtering criteria statements, but editing those entries online can be tedious. Therefore, we recommend editing complex access control lists offline within a configuration file and downloading the configuration file later via TFTP to your SmartNode device.

Creating an access control list profile and enter configuration mode

This procedure describes how to create an IP access control list and enter access control list configuration mode

Mode: Administrator execution

Step	Command	Purpose
1	node(cfg)#profile acl <i>name</i>	Creates the access control list profile <i>name</i> and enters the configuration mode for this list

name is the name by which the access list will be known. Entering this command puts you into *access control list configuration mode* where you can enter the individual statements that will make up the access control list.

Use the **no** form of this command to delete an access control list profile. You cannot delete an access control list profile if it is currently linked to an interface. When you leave the access control list configuration mode, the new settings immediately become active.

Example: Create an access control list profile

In the following example the access control list profile named WanRx is created and the shell of the access control list configuration mode is activated.

```
SN>enable
SN#configure
SN(cfg)#profile acl WanRx
SN(pf-acl)[WanRx]#
```

Adding a filter rule to the current access control list profile

The commands **permit** or **deny** are used to define an IP filter rule. This procedure describes how to create an IP access control list entry that permits access

Mode: Profile access control list

Step	Command	Purpose
1	node(pf-acl)[name]#permit ip { <i>src src-wildcard</i> any host <i>src</i> } { <i>dest dest-wildcard</i> any host <i>dest</i> } [cos <i>group</i>]	Creates an IP access of control list entry that permits access defined according to the command options

This procedure describes how to create an IP access control list entry that denies access

Mode: Profile access control list

Step	Command	Purpose
1	node(pf-acl)[name]#deny ip {src src-wildcard any host src} {dest dest-wildcard any host dest} [cos group]	Creates an IP access of control list entry that denies access defined according to the command options

Where the syntax is:

Keyword	Meaning
src	The source address to be included in the rule. An IP address in dotted-decimal-format, e.g. 64.231.1.10.
src-wildcard	A wildcard for the source address. Expressed in dotted-decimal format this value specifies which bits are significant for matching. One-bits in the wildcard indicate that the corresponding bits are ignored. An example for a valid wildcard is 0.0.0.255, which specifies a class C network.
any	Indicates that IP traffic to or from all IP addresses is to be included in the rule.
host src	The address of a single source host.
dest	The destination address to be included in the rule. An IP address in dotted-decimal-format, e.g. 64.231.1.10.
dest-wildcard	A wildcard for the destination address. See <i>src-wildcard</i>
host dest	The address of a single destination host.
cos	Optional. Specifies that packets matched by this rule belong to a certain Class of Service (CoS). For detailed description of CoS configuration refer to chapter 14, “ Link scheduler configuration ” on page 151.
group	CoS group name.

If you place a *deny ip any any* rule at the top of an access control list profile, no packets will pass regardless of the other rules you defined.

Example: Create IP access control list entries

Select the access-list profile named WanRx and create some filter rules for it.

```
SN(cfg)#profile acl WanRx
SN(pf-acl)[WanRx]#permit ip host 62.1.2.3 host 193.14.2.11 cos Urgent
SN(pf-acl)[WanRx]#permit ip 62.1.2.3 0.0.255.255 host 193.14.2.11
SN(pf-acl)[WanRx]#permit ip 97.123.111.0 0.0.0.255 host 193.14.2.11
SN(pf-acl)[WanRx]#deny ip any any
SN(pf-acl)[WanRx]#exit
SN(cfg)#
```

Adding an ICMP filter rule to the current access control list profile

The command **permit** or **deny** are used to define an ICMP filter rule. Each ICMP filter rule represents an ICMP access of control list entry.

This procedure describes how to create an ICMP access control list entry that permits access

Mode: Profile access control list

Step	Command	Purpose
1	node(pf-acl)[name]#permit icmp {src src-wildcard any host src} {dest dest-wildcard any host dest} [msg name type type type type code code] [cos group]	Creates an ICMP access of control list entry that permits access defined according to the command options

This procedure describes how to create an ICMP access control list entry that denies access

Mode: Profile access control list

Step	Command	Purpose
1	node(pf-acl)[name]#deny icmp {src src-wildcard any host src} {dest dest-wildcard any host dest} [msg name type type type type code code] [cos group]	Creates an ICMP access of control list entry that denies access defined according to the command options

Where the syntax is as following:

Keyword	Meaning
src	The source address to be included in the rule. An IP address in dotted-decimal-format, e.g. 64.231.1.10.
src-wildcard	A wildcard for the source address. Expressed in dotted-decimal format this value specifies which bits are significant for matching. One-bits in the wildcard indicate that the corresponding bits are ignored. An example for a valid wildcard is 0.0.0.255, which specifies a class C network.
any	Indicates that IP traffic to or from all IP addresses is to be included in the rule.
host src	The address of a single source host.
dest	The destination address to be included in the rule. An IP address in dotted-decimal-format, e.g. 64.231.1.10
dest-wildcard	A wildcard for the destination address. See <i>src-wildcard</i> .
host dest	The address of a single destination host.
msg name	The ICMP message name. The following are valid message names: administratively-prohibited, alternate-address, conversion-error, dod-host-prohibited, dod-net-prohibited, echo, echo-reply, general-parameter-problem, host-isolated, host-precedence-unreachable, host-redirect, host-tos-redirect, host-tos-unreachable, host-unknown, host-unreachable, information-reply, information-request, mask-reply, mask-request, mobile-redirect, net-redirect, net-tos-redirect, net-tos-unreachable, net-unreachable, network-unknown, no-room-for-option, option-missing, packet-too-big, parameter-problem, port-unreachable, precedence-unreachable, protocol-unreachable, reassembly-timeout, redirect, router-advertisement, router-solicitation, source-quench, source-route-failed, time-exceeded, timestamp-reply, timestamp-request, traceroute, ttl-exceeded, unreachable
type type	The ICMP message type. A number from 0 to 255 (inclusive)
code code	The ICMP message code. A number from 0 to 255 (inclusive)
cos	Optional. Specifies that packets matched by this rule belong to a certain Class of Service (CoS). For detailed description of CoS configuration refer to chapter 14, “ Link scheduler configuration ” on page 151.
group	CoS group name.

If you place a *deny ip any any* rule at the top of an access-list profile, no packets will pass regardless of the other rules you defined.

Example: Create ICMP access control list entries

Select the access-list profile named WanRx and create the rules to filter all ICMP echo requests (as used by the ping command).

```
SN(cfg)#profile acl WanRx
SN(pf-acl)[WanRx]#deny icmp any any type 8 code 0
SN(pf-acl)[WanRx]#exit
SN(cfg)#
```

The same effect can also be obtained by using the simpler message name option. See the following example.

```
SN(cfg)#profile acl WanRx
SN(pf-acl)[WanRX]#deny icmp any any msg echo
SN(pf-acl)[WanRX]#exit
SN(cfg)#
```

Adding a TCP, UDP or SCTP filter rule to the current access control list profile

The commands **permit** or **deny** are used to define a TCP, UDP or SCTP filter rule. Each TCP, UDP or SCTP filter rule represents a respective access of control list entry.

This procedure describes how to create a TCP, UDP or SCTP access control list entry that permits access

Mode: Profile access control list

Step	Command	Purpose
1	node(pf-acl)[name]#permit {tcp udp sctp} {src src-wildcard any host src} [{eq port gt port lt port range from to}] {dest dest-wildcard any host dest} [{eq port gt port lt port range from to}] [{cos group cos-rtp group-data group-ctrl}]	Creates a TCP, UDP or SCTP access of control list entry that permits access defined according to the command options

This procedure describes how to create a TCP, UDP or SCTP access control list entry that denies access

Mode: Profile access control list

Step	Command	Purpose
1	node(pf-acl)[name]#deny {tcp udp sctp} {src src-wildcard any host src} [{eq port gt port lt port range from to}] {dest dest-wildcard any host dest} [{eq port gt port lt port range from to}] [{cos group cos-rtp group-data group-ctrl}]	Creates a TCP, UDP or SCTP access of control list entry that denies access defined according to the command options

Where the syntax is:

Keyword	Meaning
src	The source address to be included in the rule. An IP address in dotted-decimal-format, e.g. 64.231.1.10.
src-wildcard	A wildcard for the source address. Expressed in dotted-decimal format this value specifies which bits are significant for matching. One-bits in the wildcard indicate that the corresponding bits are ignored. An example for a valid wildcard is 0.0.0.255, which specifies a class C network.
any	Indicates that IP traffic to or from all IP addresses is to be included in the rule.
host src	The address of a single source host.
eq port	Optional. Indicates that a packets port must be equal to the specified port in order to match the rule.
lt port	Optional. Indicates that a packets port must be less than the specified port in order to match the rule.
gt port	Optional. Indicates that a packets port must be greater than the specified port in order to match the rule
range from to	Optional. Indicates that a packets port must be equal or greater than the specified from port and less than the specified to port to match the rule.
dest	The destination address to be included in the rule. An IP address in dotted-decimal-format, e.g. 64.231.1.10.
dest-wildcard	A wildcard for the destination address. See <i>src-wildcard</i> .
host dest	The address of a single destination host.
cos	Optional. Specifies that packets matched by this rule belong to a certain Class of Service (CoS). For detailed description of CoS configuration refer to chapter 14, “ Link scheduler configuration ” on page 151.
cos-rtp	Optional. Specifies that the rule is intended to filter RTP/RTCP packets. In this mode you can specify different CoS groups for data packets (even port numbers) and control packets (odd port numbers). Note: this option is only valid when protocol UDP is selected.
group	CoS group name.
group-data	CoS group name for RTP data packets. Only valid when the rtp option has been specified
group-ctrl	CoS group name for RTCP control packets. Only valid when the rtp option has been specified.

Example: Create TCP, UDP or SCTP access control list entries

Select the access-list profile named WanRx and create the rules for:

Permitting any TCP traffic to host 193.14.2.10 via port 80, and permitting UDP traffic from host 62.1.2.3 to host 193.14.2.11 via any port in the range from 1024 to 2048.

```
SN(cfg)#profile acl WanRx
SN(pf-acl)[WanRx]#permit tcp any host 193.14.2.10 eq 80
SN(pf-acl)[WanRx]#permit udp host 62.1.2.3 host 193.14.2.11 range 1024 2048
SN(pf-acl)[WanRx]#exit
SN(cfg)#
```

Binding and unbinding an access control list profile to an IP interface

The command **use** is used to bind an access control list profile to an IP interface. This procedure describes how to bind an access control list profile to incoming packets on an IP interface

Mode: Profile access control list

Step	Command	Purpose
1	node(if-ip)[if-name]#use profile acl name in	Binds access control list profile name to incoming packets on IP interface <i>if-name</i>

Where the syntax is:

Keyword	Meaning
if-name	The name of the IP interface to which an access control list profile gets bound
name	The name of an access control list profile that has already been created using the profile acl command. This argument must be omitted in the no form
in	Specifies that the access control list profile applies to incoming packets on this interface.
out	Specifies that the access control list applies to outgoing packets on this interface.

The **no** form of the **use** command is used to unbind an access control list profile from an interface. When using this form the name of an access control list profile, represented by the *name* argument above, is not required. This procedure describes how to unbind an access control list profile to incoming packets on an IP interface

Mode: Interface

Step	Command	Purpose
1	node(if-ip)[if-name]#no use profile acl in	Unbinds access control list profile for incoming packets on IP interface <i>if-name</i>

Where the syntax is:

Keyword	Meaning
if-name	The name of the IP interface to which an access control list profile gets bound
in	Specifies that the access control list profile applies to incoming packets on this interface.
out	Specifies that the access control list applies to outgoing packets on this interface.

Thus for each IP interface only one incoming and outgoing access control list can be active at the same time.

Example: Bind and unbind an access control list entries to an IP interface

Bind an access control list profile to incoming packets on the interface *wan* in the IP router context.

```
SN(cfg)#context ip router
SN(cfg-ip)[router]#interface wan
SN(cfg-if)[wan]#use profile acl WanRx in
```

Unbind an access control list profile from an interface.

```
SN(cfg)#context ip router
SN(cfg-ip)[router]#interface wan
SN(cfg-if)[wan]#no use profile acl in
```

Note When unbinding an access control list profile the *name* argument is not required, since only one incoming and outgoing access control list can be active at the same time on a certain IP interface.

Displaying an access control list profile

The **show profile acl** command displays the indicated access control list profile. If no specific profile is selected all installed access control list profiles are shown. If an access control list is linked to an IP interface the number of matches for each rule is displayed. If the access control list profile is linked to more than one IP interface, it will be shown for each interface.

This procedure describes how to display a certain access control list profile

Mode: Administrator execution or any other mode, except the operator execution mode

Step	Command	Purpose
1	node#show profile acl <i>name</i>	Displays the access control list profile <i>name</i>

Example: Displaying an access control list entries

The following example shows how to display the access control list profile named WanRx.

```
SN#show profile acl WanRx
IP access-list WanRx. Linked to router/wan/in.
deny icmp any any msg echo
permit ip 62.1.2.3 0.0.255.255 host 193.14.2.11
permit ip 97.123.111.0 0.0.0.255 host 193.14.2.11
permit tcp any host 193.14.2.10 eq 80
permit udp host 62.1.2.3 host 193.14.2.11 range 1024 2048
deny ip any any
```

Debugging an access control list profile

The **debug acl** command is used to debug the access control list profiles during system operation. Use the **no** form of this command to disable any debug output.

This procedure describes how to debug the access control list profiles

Mode: Administrator execution or any other mode, except the operator execution

Step	Command	Purpose
1	node#debug acl	Enables access control list debug monitor

This procedure describes how to activate the debug level of an access control list profiles for a specific interface.

Mode: Interface

Step	Command	Purpose
1	node(cfg)#context ip router	Selects the IP router context
2	node(ctx-ip)[router]#interface if-name	Selects IP interface <i>if-name</i> for which access control list profile shall be debugged
3	node(if-ip)[if-name]#debug acl {in out} [level]	Enables access control list debug monitor with a certain debug level for the selected interface <i>if-name</i>

Where the syntax is:

Keyword	Meaning
if-name	The name of the IP interface to which an access control list profile gets bound
level	The detail level. Level 0 disables all debug output, level 7 shows all debug output.
in	Specifies that the settings for incoming packets are to be changed.
out	Specifies that the settings for outgoing packets are to be changed.

Example: Debugging access control list profiles

The following example shows how to enable debugging for incoming traffic of access control lists on interface *wan*. On level 7 all debug output is shown.

```
SN(cfg)#context ip router
SN(cfg-ip)[router]#interface wan
SN(cfg-if)[wan]#debug acl in 7
```

The following example enables the debug monitor for access control lists globally.

```
SN#debug acl
```

The following example disables the debug monitor for access control lists globally.

```
SN#no debug acl
```

Examples

Denying a specific subnet

Figure 42 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 *lan* of the SmartNode device. 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 *lan*.

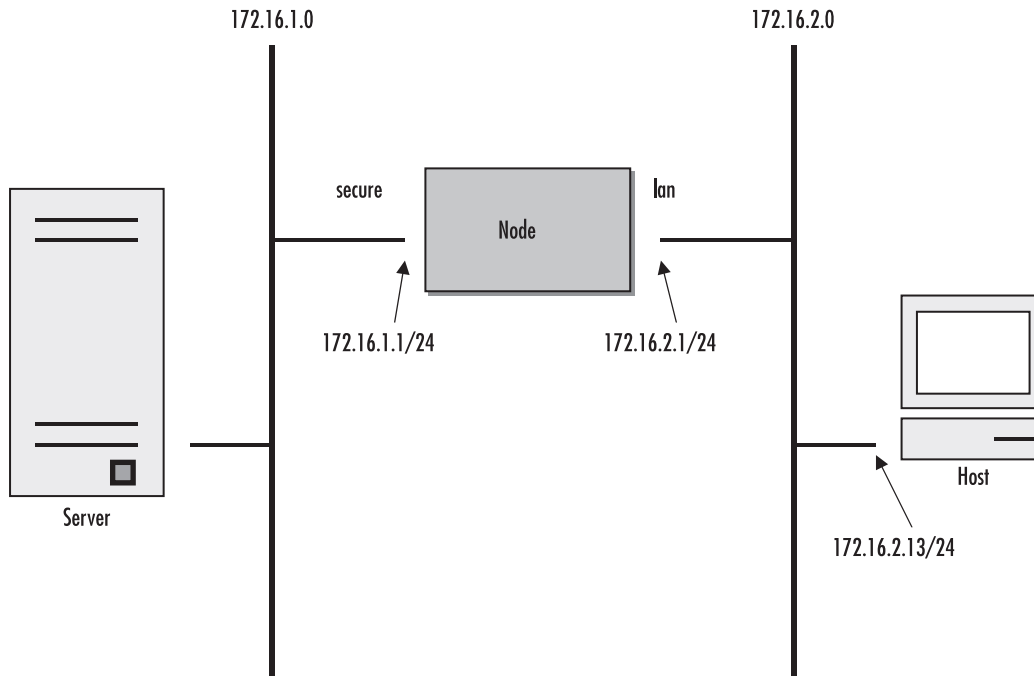


Figure 42. Deny a specific subnet on an interface

The commands that have to be entered are listed below. The commands access the SmartNode device via a Telnet session running on a host with IP address 172.16.2.13, which accesses the SmartNode via IP interface *lan*.

```
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 172.16.2.0 0.0.0.255 172.16.1.0 0.0.0.255
172.16.2.1(pf-acl)[Jamming]#permit ip any any
172.16.2.1(pf-acl)[Jamming]#exit
172.16.2.1(cfg)#context ip router
172.16.2.1(cfg-ip)[router]#interface lan
172.16.2.1(if-ip)[lan]#use profile acl Jamming in
172.16.2.1(if-ip)[lan]#exit
172.16.2.1(cfg-ip)#copy running-config startup-config
```

Chapter 19 **SNMP configuration**

Chapter contents

Introduction	218
Simple Network Management Protocol (SNMP)	218
SNMP basic components	218
SNMP basic commands	218
SNMP management information base (MIB)	219
Network management framework	219
Identification of the SmartNode 1000, 2000 and 4000 Series via SNMP	220
SNMP tools.....	221
SNMP configuration task list	221
Setting basic system information.....	222
Setting access community information	224
Setting allowed host information	225
specifying the default SNMP trap target	225
Displaying SNMP related information	226
Using the AdventNet SNMP utilities	227
Using the MibBrowser	227
Using the TrapViewer	228
Standard SNMP version 1 traps.....	230
SNMP interface traps	232

Introduction

This chapter provides overview information about Simple Network Management Protocol (SNMP) and describes the tasks used to configure those of its features supported by SmartWare.

This chapter includes the following sections:

- Simple Network Management Protocol (SNMP)
- SNMP tools (see [page 221](#))
- SNMP configuration task list (see [page 221](#))
- Using the AdventNet SNMP utilities (see [page 227](#))
- Standard SNMP version 1 traps (see [page 230](#))

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.

Two versions of SNMP exist: SNMP version 1 (SNMPv1) and SNMP version 2 (SNMPv2). Both versions have a number of features in common, but SNMPv2 offers enhancements, such as additional protocol operations. Standardization of yet another version of SNMP—SNMP version 3 (SNMPv3)—is pending. This chapter provides general descriptions of the SNMP version 1 and 2 protocol operations. Be aware that the SNMP agent running in SmartWare is SNMP version 1 (SNMPv1) compliant.

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 node 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 the SmartNode 1000, 2000 and 4000 Series via SNMP

All models of the Patton SmartNode 1000, 2000 or 4000 series devices have their unambiguous value assigned sysObjectID (.iso.org.dod.internet.mgmt.mib-2.system.sysObjectID) object.

Table 11. The returned value when reading the sysObjectID object for each SmartNode model

SmartNode 4112	.iso.org.dod.internet.private.enterprises.patton.products.sn4000.sn4112 1.3.6.1.4.1. 5349.2.4.5.5
SmartNode 4114	.iso.org.dod.internet.private.enterprises.patton.products.sn4000.sn4114 1.3.6.1.4.1. 5349.2.4.5.6
SmartNode 4116	.iso.org.dod.internet.private.enterprises.patton.products.sn4000.sn4116 1.3.6.1.4.1. 5349.2.4.5.7
SmartNode 4118	.iso.org.dod.internet.private.enterprises.patton.products.sn4000.sn4118 1.3.6.1.4.1. 5349.2.4.5.8
SmartNode 4522	.iso.org.dod.internet.private.enterprises.patton.products.sn4000.sn4522 1.3.6.1.4.1. 5349.2.4.5.2
SmartNode 4524	.iso.org.dod.internet.private.enterprises.patton.products.sn4000.sn4524 1.3.6.1.4.1. 5349.2.4.5.1
SmartNode 4526	.iso.org.dod.internet.private.enterprises.patton.products.sn4000.sn4526 1.3.6.1.4.1. 5349.2.4.5.3
SmartNode 4528	.iso.org.dod.internet.private.enterprises.patton.products.sn4000.sn4528 1.3.6.1.4.1. 5349.2.4.5.4

Table 12. SmartNode Models and their Unique sysObjectID

SmartNode Model	SysObjectID
SmartNode 1200	.iso.org.dod.internet.private.enterprises.patton.products.sn1200 1.3.6.1.4.1. 5349.2.4.1
SmartNode 1400	.iso.org.dod.internet.private.enterprises.patton.products.sn1400 1.3.6.1.4.1. 5349.2.4.2
SmartNode 2300	.iso.org.dod.internet.private.enterprises.patton.products.sn2300 1.3.6.1.4.1. 5349.2.4.3
SmartNode 2400	.iso.org.dod.internet.private.enterprises.patton.products.sn2400 1.3.6.1.4.1. 5349.2.4.4
SmartNode 4112	.iso.org.dod.internet.private.enterprises.patton.products.sn4000.sn4112 1.3.6.1.4.1. 5349.2.4.5.5
SmartNode 4114	.iso.org.dod.internet.private.enterprises.patton.products.sn4000.sn4114 1.3.6.1.4.1. 5349.2.4.5.6
SmartNode 4116	.iso.org.dod.internet.private.enterprises.patton.products.sn4000.sn4116 1.3.6.1.4.1. 5349.2.4.5.7

Table 12. SmartNode Models and their Unique sysObjectID (Continued)

SmartNode Model	SysObjectID
SmartNode 4118	.iso.org.dod.internet.private.enterprises.patton.products.sn4000.sn4118 1.3.6.1.4.1. 5349.2.4.5.8
SmartNode 4522	.iso.org.dod.internet.private.enterprises.patton.products.sn4000.sn4522 1.3.6.1.4.1. 5349.2.4.5.2
SmartNode 4524	.iso.org.dod.internet.private.enterprises.patton.products.sn4000.sn4524 1.3.6.1.4.1. 5349.2.4.5.1
SmartNode 4526	.iso.org.dod.internet.private.enterprises.patton.products.sn4000.sn4526 1.3.6.1.4.1. 5349.2.4.5.3
SmartNode 4528	.iso.org.dod.internet.private.enterprises.patton.products.sn4000.sn4528 1.3.6.1.4.1. 5349.2.4.5.4

According to table 12, an SNMP get request to *.iso.org.dod.internet.mgmt.mib-2.system.sysObjectID* of a SmartNode 1200 device reads out a numeric OID of *1.3.6.1.4.1. 5349.2.2.1*, which represents a SmartNode 1200 device. The mapping of the sysObjectID to each of the SmartNode model is realized with the SmartNode product identification MIB.



The SNMP agent running in SmartWare is SNMP version 1 (SNMPv1) compliant. SNMP version 2 (SNMPv2) and SNMP version 3 (SNMPv3) are not currently supported.

SNMP tools

Patton recommends the AdventNet MibBrowser, TrapViewer and other SNMP tools. Check the AdventNet Web server at <http://www.adventnet.com> for latest releases.

Refer to section “[Using the AdventNet SNMP utilities](#)” on page 227 for more detailed information on how to use these tools together with SmartNode 1000, 2000 or 4000 series devices.

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 222](#))
- Setting access community information (required) (see [page 224](#))
- Setting allowed host information (required) (see [page 225](#))
- Specifying the default SNMP trap target (optional) (see [page 225](#))
- Displaying SNMP related information (optional) (see [page 226](#))

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.

On the SmartNode 1000, 2000 or 4000 series device appropriate values should be set for the following MIB-II system group objects:

- sysContact
- sysLocation
- sysName

The system sysContact object is used to define the contact person for this managed SmartNode, together with information on how to contact that person.

Assigning explanatory location information to describe the system physical location of your SmartNode (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	node(cfg)#system contact <i>name</i>	Sets the contact persons name
2	node(cfg)#system location <i>location</i>	Sets the system location
3	node(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.

After setting a hostname the prompt of the command line, normally representing the IP address of the Ethernet port over which the SmartNode is accessed via Telnet, is replaced with the hostname.

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 43](#).

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.

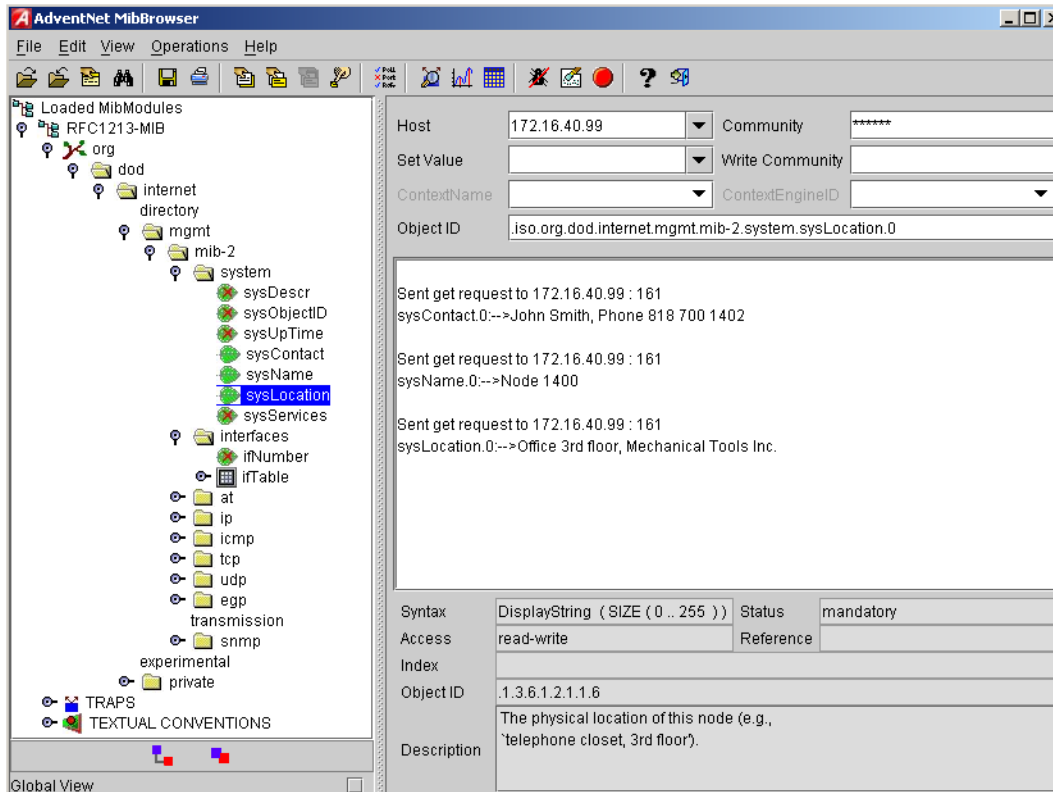


Figure 43. AdventNet MibBrowser displaying some of the System Group objects

Example: Setting the system group objects

In the following example the system information is set for later access via SNMP. See [figure 43](#) for a typical MIB browser application accessing these MIB-II system group objects representing the system information.

```
SN>enable
SN#configure
SN(cfg)#system contact "Lorenz Born, Phone 533"
SN(cfg)#system location "Office, 3rd floor, Patton Electronics Co."
SN(cfg)#system hostname "SN2300-01"
SN2300-01(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.

By default SNMP uses the default communities *public* and *private*. You probably do not want to use those, as they are the first things an intruder will look for. 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 community name { ro rw }</code>	Configures the SNMP community name with read-only or read/write access

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 running on the respective SmartNode 1000, 2000 or 4000 series device.

```
SN2300-01(cfg)#snmp community public ro
SN2300-01(cfg)#snmp community Not4evEryOne rw
```

Note If no community is set on your SmartNode 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 node, it explicitly needs the right to access the SNMP agent on the respective SmartNode 1000, 2000 or 4000 series device. Therefore a host needs an entry on a SmartNode 1000, 2000 or 4000 series device, 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	<code>node(cfg)#snmp host IP-address-of-node security-name community</code>	Configures a host that with IP address <i>IP-address-of-node</i> can access the MIB of this SmartNode 1000, 2000 or 4000 series device, 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 of this SmartNode 2000 series device using community `public` as security name.

```
SN2300-01(cfg)#snmp host 172.16.224.45 security-name public
```

specifying the default SNMP trap target

An SNMP trap is a message that the SNMP agent running on a SmartNode 1000, 2000 or 4000 series device 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 running on a SmartNode 1000, 2000 or 4000 series device 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	node(cfg)#snmp target <i>IP-address-of-node</i> security-name <i>community</i>	Configures a SNMP trap target with IP-address-of-hostname <i>node</i> that receives trap messages of this SmartNode 1000, 2000 or 4000 series device, 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.

```
SN2300-01(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 running on a SmartNode 1000, 2000 or 4000 series device.

This procedure describes how to display information and configuration settings for SNMP

Mode: Configure

Step	Command	Purpose
1	node(cfg)#show snmp	Displays information and configuration settings for SNMP

Example: Displaying SNMP related information

This example shows how to display SNMP configuration information.

```
SN2300-01(cfg)#show snmp
```

```
SNMP Information:
```

```
hostname : SN2300-01
location : Office, 3rd floor, Patton Electronics Co.
contact  : Lorenz Born, Phone 533
```

```
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 AdventNet SNMP utilities

The AdventNet 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 tools can communicate and interact with any SNMP enabled device, such as a SmartNode 1000, 2000 or 4000 series device. The following tools are the most useful part of the product for SmartNode 1000, 2000 or 4000 series device management:

- **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 AdventNet MibBrowser 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 44 depicts the primary window of the AdventNet 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 node. The AdventNet MibBrowser allows loading additional MIB files in the text format, e.g. the Patton Electronics Co. enterprise specific MIBs used for SmartNode 1000, 2000 or 4000 series device management.

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 → Display).
- The other way of altering the view is through the general settings panel in the settings menu item in the edit menu. (Edit → Settings)

- The same can be done through clicking the MibBrowser settings button on the toolbar. See [figure 44](#).



Figure 44. AdventNet 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 running on a SmartNode 1000, 2000 or 4000 series device. The TrapViewer can listen to one or more port at a time and the traps can be sent from any host. SmartNode 1000, 2000 or 4000 series device send their traps to the SNMP standard port 162.

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 227. [Figure 45](#) is a screen shot of the TrapViewer.

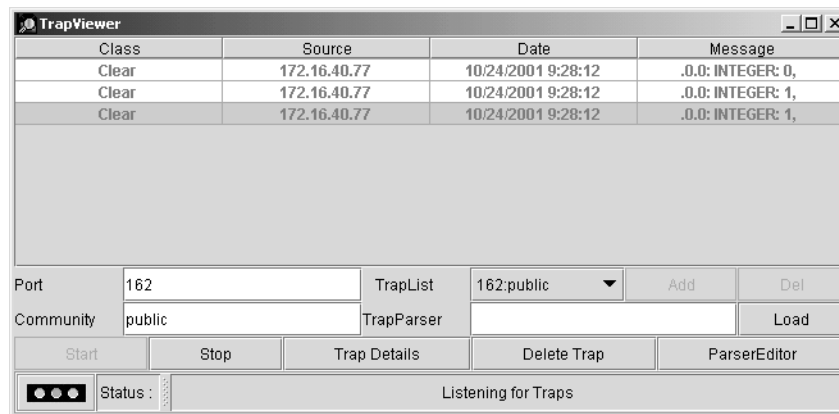


Figure 45. AdventNet TrapViewer displaying received traps

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 of your SmartNode 1000, 2000 or 4000 series device.
- 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 nodes 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 46 show the screen of such a trap details window.

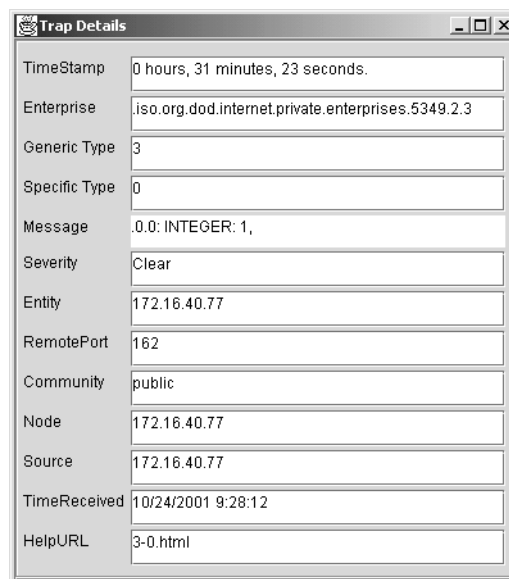


Figure 46. AdventNet Trap Details window of TrapViewer

The various details available in the Trap Details window are listed in table 13:

Table 13. 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-eggNeighborLoss. 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.
Node	Source
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 AdventNet SNMP Utilities documentation for a detailed description of the parser editor configuration and its use.

Standard SNMP version 1 traps

SmartWare application software supports the following standard SNMP version 1 traps. 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 implementa-
tion 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 SmartNode 1000, 2000 or 4000 series device sends a warmStart trap message if any trap target host is defined.

SNMP interface traps

The SmartNode sends Interface Traps (*linkUp*, *linkDown*) when the status of logical or physical interfaces change. Logical interfaces are interfaces defined in the IP context (IP interfaces) and interfaces defined in the CS context (PSTN, SIP, and H323 interfaces). Physical interfaces are ports in the SmartNode terminology (Ethernet, ISDN, and Serial Ports).

The SmartNode assigns an index to each interface (ifIndex) in order to identify the Interface Traps. These assignments depend on the hardware and software configurations. The command **show snmp-if-alias-mapping** displays the relations between the indexes and the interfaces. It also lists the status of the interfaces.

```
SN(cfg)#show snmp-if-alias-mapping
```

ifIndex :	Interface Name	(Interface Type)	Interface Status
1 :	h323_60	(H323)	up
2 :	h323_30	(H323)	up
3 :	isdn20	(PSTN)	up
4 :	ETH00	(ethernet-csmacd)	up
5 :	ETH01	(ethernet-csmacd)	up
6 :	eth00	(IP)	up
7 :	eth01	(IP)	up
8 :	ISDN20	(pstn)	down

Interface names in capital letters denote physical interfaces and in small letters logical interfaces.

The SmartNode adds an entry to event log for each Interface Traps it sends:

```
SN(cfg)#show log
```

```
...
2002-09-06T14:54:35 : LOGINFO : Link up on interface h323_60.
2002-09-06T14:54:35 : LOGINFO : Link up on interface h323_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 20 **SNTP client configuration**

Chapter contents

Introduction	234
SNTP client configuration task list	234
Selecting SNTP time servers	235
Defining SNTP client operating mode	235
Defining SNTP local UDP port	236
Enabling and disabling the SNTP client	237
Defining SNTP client poll interval	237
Defining SNTP client constant offset to GMT	237
Defining the SNTP client anycast address	238
Enabling and disabling local clock offset compensation	239
Showing SNTP client related information	240
Debugging SNTP client operation	240
Recommended public SNTP time servers.....	241
NIST Internet time service	241
Other public NTP primary (stratum 1) time servers	242
Additional information on NTP and a list of other NTP servers	243
Recommended RFC	243

Introduction

This chapter describes how to configure Simple Network Time Protocol (SNTP) client, it includes the following sections:

- SNTP client configuration task list
- Recommended Public SNTP Time Servers (see [page 241](#))

The Simple Network Time Protocol (SNTP) is an adaptation of the Network Time Protocol (NTP) that is used to synchronize computer clocks in the Internet. SNTP can be used when the ultimate performance of the full NTP implementation is not needed. SNTP is described in RFC-2030, “Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI”.

SNTP typically provides time within 100 milliseconds of the accurate time, but it does not provide the complex filtering and statistical mechanisms of NTP. In addition, SNTP does not authenticate traffic, although you can configure extended access lists to provide some protection. An SNTP client is more vulnerable to misbehaving servers than an NTP client and should only be used in situations where strong authentication is not required.

SNTP client configuration task list

To configure an SNTP client, perform the tasks described in the following sections. The tasks in the first four sections are required; the tasks in the remaining sections are optional, but might be required for your application.

- Selecting SNTP time servers (see [page 235](#))
- Defining SNTP client operating mode (see [page 235](#))
- Defining SNTP local UDP port (see [page 236](#))
- Enabling and disabling the SNTP client (see [page 237](#))
- Defining the SNTP client anycast address (see [page 238](#))
- Defining SNTP client constant offset to GMT (see [page 237](#))
- Enabling and disabling local clock offset compensation (see [page 239](#))
- Defining SNTP client poll interval (see [page 237](#))
- Showing SNTP client related information (see [page 240](#))
- Debugging SNTP client operation (see [page 240](#))

Selecting SNTP time servers

This procedure describes how to select a primary and secondary SNTP time server

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg)#sntp-client server primary host</code>	Enter the SNTP primary server IP address or hostname
2	<code>node(cfg)#sntp-client server secondary host</code>	Enter the SNTP secondary server IP address or hostname

Example: Selecting SNTP time servers

In the following example an internal SNTP time server (172.16.1.10) is selected as primary and utcnist.colorado.edu (128.138.140.44) as secondary SNTP time server.

```
SN(cfg)#sntp-client server primary 172.16.1.10
SN(cfg)#sntp-client server secondary 128.138.140.44
```

Defining SNTP client operating mode

A SNTP client can operate in multicast mode, unicast mode or anycast mode:

- In unicast mode (point to point), the client sends a request to a designated server at its unicast address and expects a reply from which it can determine the time and, optionally, the roundtrip delay and local clock offset relative to the server.
- In anycast mode (multipoint to point), the client sends a request to a designated local broadcast or multicast group address and expects a reply from one or more anycast servers.
- In multicast mode (point to multipoint), the client sends no request and waits for a broadcast from a designated multicast server.

Note Unicast mode is the default SNTP client operating mode.

This procedure describes how to configure the SNTP client operating mode

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg)#sntp-client operating-mode {unicast anycast multicast}</code>	Configures the SNTP client operating mode to unicast, anycast or multicast mode

Note When selecting the anycast operating-mode you have to define the IP address where the anycast request is sent. Refer to section “[Defining the SNTP client anycast address](#)” on page 238 for more details.

Example: Configuring SNTP client operating mode

Configures the SNTP client operating mode to unicast operation

```
SN(cfg)#sntp-client operating-mode unicast
```

Configures the SNTP client operating mode to anycast operation

```
SN(cfg)#sntp-client operating-mode anycast
```

Configures the SNTP client operating mode to multicast operation

```
SN(cfg)#sntp-client operating-mode multicast
```

Defining SNTP local UDP port

The communication between an SNTP client and its the primary or secondary SNTP time server uses UDP. The UDP port number assigned to SNTP is 123, which should be used in both the source port (on the Smart-Node) and destination port (on SNTP time server) fields in the UDP header. The local port number, which the SNTP client uses to contact the primary or secondary SNTP time server in unicast mode, has to be defined.

Note The local port number setting is used when contacting the SNTP time server. The SNTP time server will send its reply to the SNTP client (Smart-Node) using the same port number as used in the request. The local port number is set to 123 by default.

This procedure describes how to define the local port number, which uses the SNTP client to contact the SNTP time server, unicast mode

Mode: Configure

Step	Command	Purpose
1	node(cfg)# sntp-client local-port <i>number</i>	Specifies the SNTP local UDP port number. The port number can be defined in the range from 1 to 65535. The UDP port number assigned to SNTP is 123.

Example: Defining the local UDP port for SNTP

Configures the SNTP client UDP port number to 123

```
SN(cfg)#sntp-client local-port 123
```

Enabling and disabling the SNTP client

The SNTP client is disabled by default and has to be enabled if clock synchronization shall be used. This procedure describes how to enable or disable the SNTP client

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg)#[no] sntp-client</code>	Enables the SNTP client operation. Using the no command syntax disables this feature.

Example: Enabling the SNTP client operation

```
SN(cfg)#sntp-client
```

Example: Disabling the SNTP client operation

```
SN(cfg)#no sntp-client
```

Defining SNTP client poll interval

Specifies the seconds between each SNTP client request in unicast or anycast mode.

This SNTP client poll interval can be defined to be within the range from 1 to 4'294'967'295. The default value is 60 seconds.

This procedure describes how to set the SNTP client poll interval

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg)#sntp-client poll-interval value</code>	Sets the SNTP client poll interval to value seconds

Example: Setting the SNTP client poll interval

In the following example the SNTP client poll interval is set to 30 seconds.

```
SN(cfg)#sntp-client poll-interval 30
```

Defining SNTP client constant offset to GMT

Setting the offset of the SmartNode 1000, 2000 or 4000 series device local time zone from Greenwich Mean Time is required if the local time shall be used for time dependent routing decisions or other reasons. Greenwich Mean Time (GMT) is also known as Zulu Time and Universal Time Coordinated (UTC), refer to <http://greenwichmeantime.com/> for more details and information about your time zone and offset to GMT.

Note Be aware that summertime offset is not automatically adjusted!

This procedure describes how to set the SNTP client local time zone offset from GMT

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg)#sntp-client gmt-offset offset</code>	Specifies the SNTP client constant offset from GMT, where offset is + or – followed by hh:mm:ss, with a range from –24:00:00 to +24:00:00

Example: Setting SNTP client local time zone offset from GMT

In the following example the SNTP client local time zone offset is set to +2 hours ahead of GMT, e.g. for Switzerland during Summer Time. Be aware that a space follows the + or – sign before the time offset is entered.

```
SN>enable
SN#configure
SN(cfg)#sntp-client gmt-offset + 02:00:00
```

There is a short form notation supported as shown in the following example.

```
SN(cfg)#sntp-client gmt-offset + 2
```

Defining the SNTP client anycast address

Anycast mode is designed for use with a set of cooperating servers whose addresses are not known beforehand by the SmartNode. An anycast client (SmartNode) sends a request to the designated local broadcast or multicast group address as described below. For this purpose, the NTP multicast group address assigned by the IANA is used. One or more anycast servers listen on the designated local broadcast address or multicast group address. Each anycast server, upon receiving a request, sends a unicast reply message to the originating client. The client then binds to the first such message received and continues operation in unicast mode. Subsequent replies from other anycast servers are ignored.

In anycast mode, the SmartNode sends a request to a designated local broadcast or multicast group address and expects a reply from one or more anycast servers. The SmartNode uses the first reply received to establish the particular server for subsequent unicast operations. Later replies from this server (duplicates) or any other server are ignored.

Other than the selection of address in the request, the operations of anycast and unicast clients are identical.

This procedure describes how to set local broadcast address or multicast group address to which the anycast request is sent

Mode: Configure

Step	Command	Purpose
1	node(cfg)#sntp-client anycast-address <i>ip-address</i> {port port-number}	Set the anycast-address to <i>ip-address</i> a designated local broadcast or multicast group address to which a request is sent. In addition an explicit SNTP server <i>port-number</i> in the range from 1 to 65535 can be defined or the argument port is selected, which sets the value for port to 123. If none of the optional argument is used the value for port is set to 123.

Note This command is only relevant in anycast operating-mode.

Example: SNTP client anycast address

In the following example anycast requests are sent to SNTP server at IP address 132.163.4.101 using port 123 of the SNTP server.

```
SN(cfg)#sntp-client anycast-address 132.163.4.101 port
```

Enabling and disabling local clock offset compensation

The Simple Network Time Protocol (SNTP) Version 4 is an adaptation of the Network Time Protocol (NTP) that is used to synchronize computer clocks in the Internet. While not necessary in a conforming SNTP client, in unicast and anycast modes it is highly recommended that the transmit timestamp in the request is set to the time of day according to the client clock in NTP timestamp format. This allows a simple calculation to determine the propagation delay between the server and client and to align the local clock generally within a few tens of milliseconds relative to the server. In addition, this provides a simple method to verify that the server reply is in fact a legitimate response to the specific client request and to avoid replays.

In multicast mode, the client has no information available to calculate the propagation delay or to determine the validity of the server unless the NTP authentication scheme is used.

This procedure describes how to enable or disable the compensation for local clock offset.

Mode: Configure

Step	Command	Purpose
1	node(cfg)#[no] sntp-client local-clock-offset	Enables the SNTP client's compensation for local clock offset. Using the no command syntax disables this feature.

Example: Enabling the SNTP client root delay compensation

```
SN(cfg)#sntp-client root-delay-compensation
```

Example: Disabling the SNTP client root delay compensation

```
SN(cfg)#no sntp-client root-delay-compensation
```

Showing SNTP client related information

During set-up and operation of the SNTP client, displaying the information and status of the SNTP client is very useful.

This procedure describes how to display information and status of the SNTP client

Mode: Configure

Step	Command	Purpose
1	node(cfg)#show sntp-client	Displays information and status of the SNTP client

Example: Showing SNTP client related information

```
SN(cfg)#show sntp-client
-----
SNTP client      enabled
Operating mode   unicast
Local port       123
Primary server   172.16.1.10:123 v4
Secondary server 128.138.140.44:123 v4
Anycast address  224.0.1.1:123
Poll interval    30sec
Local clock offset disabled
GMT offset       +2:00:00
-----
```

Debugging SNTP client operation

During setup and operation, debugging the behavior of the SNTP client is very useful.

Note The debug sntp client is only available in superuser mode.

This procedure describes how to enable or disable debugging

Mode: Configure

Step	Command	Purpose
1	node(cfg)#debug sntp client	Enables and disables SNTP debug monitor. Using the no command syntax disables this feature.

Example: Enable the SNTP debug monitor

The following example shows how to enable the SNTP debug monitor and some typical debug information.

```
SN(cfg)#debug sntp client
SN(cfg)#14:44:21 SNTP > SNTP message sent with Timestamp: 2001-10-26T14:44:21
14:44:21 SNTP > SNTP message received:
-----
Server:      172.16.1.10:123 v4
Stratum:    2
Time:       2001-10-26T12:44:21
InternetTime: 20010926@530
-----
14:44:21 SNTP > Set the system time to 2001-10-26T14:44:21
14:44:51 SNTP > SNTP message sent with Timestamp: 2001-10-26T14:44:51
14:45:21 SNTP > SNTP message sent with Timestamp: 2001-10-26T14:45:21
14:45:51 SNTP > SNTP message sent with Timestamp: 2001-10-26T14:45:51
14:46:21 SNTP > SNTP message sent with Timestamp: 2001-10-26T14:46:21
14:46:51 SNTP > SNTP message sent with Timestamp: 2001-10-26T14:46:51
```

Example: Disable the SNTP debug monitor

The following example shows how to disable the SNTP debug monitor and end any debug information.

```
SN(cfg)#no debug sntp client
```

Recommended public SNTP time servers

NIST Internet time service

The National Institute of Standards and Technology (NIST) Internet Time Service allows users to synchronize computer clocks via the Internet. The time information provided by the service is directly traceable to UTC. Table 14 contains information about all of the time servers operated by NIST. Please note that while NIST makes every effort to ensure that the names of the servers are correct, NIST only controls the names of the nist.gov servers. It is probably safest to use the IP addresses instead of the domain names.

Table 14. Time servers operated by NIST

Server Name	IP Address	Note	Location
nist1.aol-va.truetime.com	205.188.185.33	2	DC/Virginia
utcnist.colorado.edu	128.138.140.44	2	Colorado
nist1.aol-ca.truetime.com	207.200.81.113	2	California
nist1-dc.glassey.com	216.200.93.8	2	DC/Virginia
nist1.datum.com	63.149.208.50	2	California
nist1-ny.glassey.com	208.184.49.9	2	New York City
nist1-sj.glassey.com	207.126.103.204	2	California
time-a.nist.gov	129.6.15.28	1	Maryland
time-b.nist.gov	129.6.15.29	1	Maryland
time-a.timefreq.blrdoc.gov	132.163.4.101	1, 4	Colorado
time-b.timefreq.blrdoc.gov	132.163.4.102	1	Colorado
time-c.timefreq.blrdoc.gov	132.163.4.103	1	Colorado

Table 14. Time servers operated by NIST (Continued)

Server Name	IP Address	Note	Location
time-d.timefreq.bldrdoc.gov	132.163.4.104	3	Colorado
time.nist.gov	192.43.244.18	1	Colorado
time-nw.nist.gov	131.107.1.10	1	Washington

Legend

1. Heavily loaded and not recommended for new users.
2. Recommended for new users.
3. Used for testing only. Not for general users.
4. Does not support anonymous ftp connections.

For more information about NIST Internet Time Service (ITS) check their web server at <http://www.boulder.nist.gov/timefreq/service/its.htm>

Other public NTP primary (stratum 1) time servers

Switzerland

- swisstime.ethz.ch (129.132.2.21)
- Location: Integrated Systems Laboratory, Swiss Fed. Inst. of Technology, CH 8092 Zurich, Switzerland
- Geographic Coordinates: 47:23N, 8:32E
- Synchronization: NTP primary (DCF77 clock), Sun-4/SunOS 4.1.4
- Service Area: Switzerland/Europe
- Access Policy: open access
- Contact: Christoph Wicki (time@iis.ee.ethz.ch)

Germany

- DE ntp0.fau.de (131.188.34.75)
- Location: University Erlangen-Nuernberg, D-91058 Erlangen, FRG
- Geographic Coordinates: 49.573N 11.028E (from Meinberg GPS 166)
- Synchronization: NTP V3 primary (GPS receiver (<<1us)), Sun SS12/Unix SunOS 5.6
- Service Area: Germany/Europe
- Access Policy: open access, pick one of ntp{0,1,2}.fau.de
- Contact: The Timekeepers (time@informatik.uni-erlangen.de)

Note IP addresses are subject to change; please use DNS

- DE ntp1.fau.de (131.188.34.45)
- Location: University Erlangen-Nuernberg, D-91058 Erlangen, FRG
- Geographic Coordinates: 49.573N 11.028E (from Meinberg GPS 166)

- Synchronization: NTP V3 primary (DCF77 PZF receiver (<50us)), Sun E3000 SunOS 5.6
- Service Area: Germany/Europe
- Access Policy: open access, pick one of ntp{0,1,2}.fau.de
- Contact: The Timekeepers (time@informatik.uni-erlangen.de)

Note IP addresses are subject to change; please use DNS

- DE ntps1-0.cs.tu-berlin.de (130.149.17.21)
- Location: Technische Universitaet Berlin, D-10587 Berlin, FRG
- Geographic Coordinates: 52.518N 13.326E
- Synchronization: NTP V3 primary (Meinberg GPS 166), Sun 4/65 SunOS4.1.3
- Service Area: Germany/Europe
- Access Policy: open access
- Contact: Gerard Gschwind (gg@cs.tu-berlin.de)

Additional information on NTP and a list of other NTP servers

The University of Delaware hosts a World Wide Web site that provides additional information on NTP and a list of other NTP servers that are publicly available around the world. In many cases, Internet service providers, universities, and other institutions also provide NTP servers for their own communities. NTP servers other than the NIST NTP servers (listed above) may or may not be of comparable accuracy, and may or may not satisfy traceability requirements. For more information, please see <http://www.eecis.udel.edu/~ntp/>.

Recommended RFC

RFC2030 “Simple Network Time Protocol (SNTP) Version 4”, is available from the Web server at <http://www.faqs.org/rfcs/rfc2030.html>.

Chapter 21 **DHCP configuration**

Chapter contents

Introduction	246
DHCP-client configuration tasks.....	247
Enable DHCP-client on an IP interface	247
Release or renew a DHCP lease manually (advanced)	248
Get debug output from DHCP-client	249
DHCP-server configuration tasks	250
Configure DHCP-server profiles	250
Use DHCP-server profiles and enable the DHCP-server	252
Check DHCP-server configuration and status	253
Get debug output from the DHCP-server	253

Introduction

This chapter provides an overview of the Dynamic Host Configuration Control Protocol (DHCP) and describes the tasks involved in their configuration. This chapter includes the following sections:

- DHCP-client configuration tasks (see [page 247](#))
- DHCP-server configuration tasks (see [page 250](#))

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 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.

In SmartWare, a DHCP-client and a DHCP-server are implemented. The DHCP-client gets IP address and configuration information from a DHCP-server on the WAN side of the SmartNode. The DHCP-server pro-

vides other clients on the LAN side with IP addresses and other configuration information. DHCP-server and DHCP-client are illustrated in [figure 47](#).

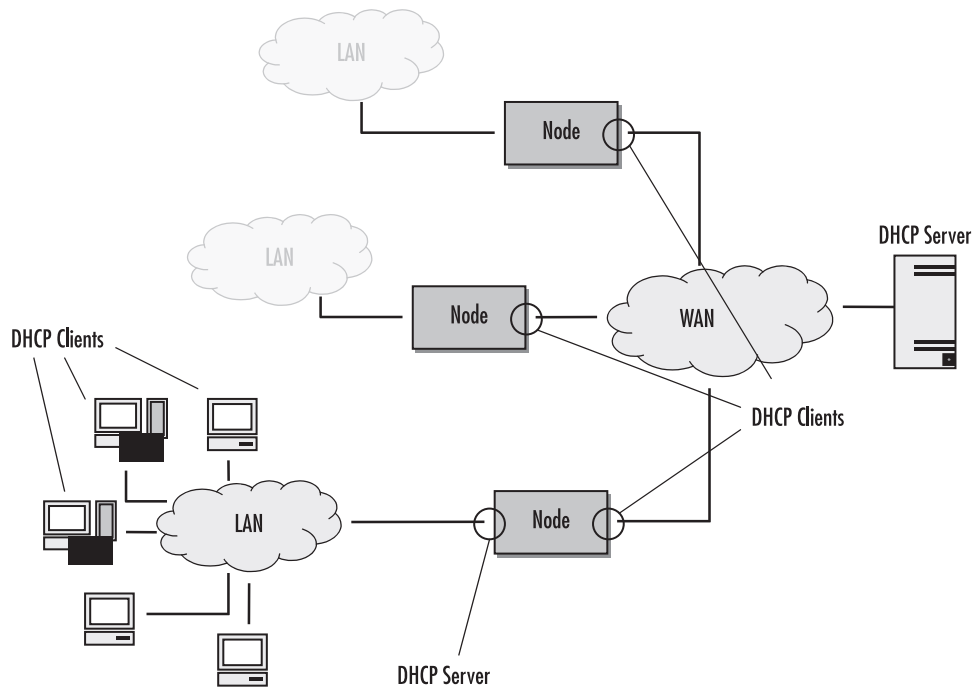


Figure 47. DHCP-client and DHCP-server on the SmartNode

DHCP-client configuration tasks

To configure the SmartNode as DHCP-client perform the steps mentioned below.

- Enable DHCP-client on an IP interface
- Release or renew a DHCP lease manually (advanced) (see [page 248](#))
- Get debug output from DHCP-client (see [page 249](#))
- Configure DHCP agent

Enable DHCP-client on an IP interface

On every created IP interface a DHCP-client could be enabled. If enabled, the SmartNode gets the IP address for this interface from a DHCP-server. Additionally other configuration information is received for this IP

interface, e.g. the default gateway, DNS server IP addresses, etc. To enable the DHCP-client on an IP interface perform the steps described below.

Mode: context IP

Step	Command	Purpose
1	<code>node(ctx-ip)[router]#interface name</code>	Creates an IP interface with name <i>name</i> and enters 'configure' configuration mode
2	<code>node(if-ip)[name]#ipaddress dhcp</code>	Enables the DHCP-client on this IP interface. (See note)
3	<code>node(if-ip)[name]#show dhcp-client</code>	Displays status information about the DHCP-client. For example, default gateway, lease expire time, etc.

Note If you are connected to the SmartNode by Telnet over the IP interface on which you enable the DHCP-client, the connection is lost after entering the command 'ipaddress dhcp'. You need to know the new IP address distributed from the DHCP-server to connect to the SmartNode again!

Example: Enable DHCP-client on an IP interface

```
SN(cfg)#context ip
SN(ctx-ip)[router]#interface eth0
SN(if-ip)[eth0]#ipaddress dhcp
SN(if-ip)[eth0]#show dhcp-client
-----
Context:                router
Name:                   eth0
IpAddress:              172.16.224.102 255.255.0.0
Default gateway:       172.16.1.10
Domain Name:           pacific
DNS:                   172.16.1.10
                       146.228.10.16
Next Server Ip:        172.16.1.10
DHCP Server:           172.16.1.10
Lease obtained:        2001-01-01T01:03:51
Lease expires:         2001-01-01T09:03:51
State:                 Bound
```

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.

This procedure describes how to release and renew the DHCP lease

Mode: interface

Step	Command	Purpose
1	node(if-ip)[name]#dhcp-client release	Releases DHCP lease. (See note)
2	node(if-ip)[name]#dhcp-client renew	Gets a new DHCP lease from the DHCP-server

Note If you are connected to the SmartNode by Telnet over the IP interface on which you release the DHCP lease, the connection is lost after entering the command **dhcp-client release**. You need an other way (e.g. a serial connection) to connect to the SmartNode again and to enter the command **dhcp-client renew**!

Get debug output from DHCP-client

This procedure describes how to enable/disable DHCP-client debug monitor

Mode: Any

Step	Command	Purpose
1	node(if-ip)[name]#[no] debug dhcp-client	Enables/disables the DHCP-client debug monitor

Example: Enable DHCP debug monitor

This example shows how to enable the DHCP-client debug monitor and the debug output of the command **dhcp-client release** and **dhcp-client renew**.

```
SN(cfg)#context ip
SN(ctx-ip)[router]#interface eth0
SN(if-ip)[eth0]#debug dhcp-client
SN(if-ip)[eth0]#dhcp-client release
01:12:28  DHCPC > router/eth0 (Rels): Unicasting DHCP release (xid 490cb56b, secs
1).
01:12:29  DHCPC > router/eth0 (Rels): Shutting down.
01:12:29  DHCPC > router/eth0 (Rels): Tearing down IP interface
2001-01-01T01:12:30 : LOGINFO      : Link down on interface eth0.
2001-01-01T01:12:30 : LOGINFO      : Link up on interface eth0.

SN(if-ip)[eth0]#dhcp-client renew
01:17:46  DHCPC > router/eth0 (Init): Tearing down IP interface
01:17:46  DHCPC > router/eth0 (Init): Broadcasting DHCP discover (xid 0f839e56, secs
0).
01:17:46  DHCPC > router/eth0 (Init):  Requesting IP address 172.16.224.102
01:17:47  DHCPC > router/eth0 (Slct): Got offer from 172.16.1.10 for IP
172.16.224.102
01:17:47  DHCPC > router/eth0 (Slct): Selected offer for 172.16.224.102
01:17:47  DHCPC > router/eth0 (Slct): Broadcasting DHCP request (select) (xid
6ff42c38, secs 1).
2001-01-01T01:17:47 : LOGINFO      : router/eth0 (Rqst): Got DHCP lease for
172.16.224.102
01:17:47  DHCPC > router/eth0 (Rqst): DHCP ACK received.
01:17:47  DHCPC > router/eth0 (Rqst): Lease is valid for 28800 seconds
```

```
01:17:47 DHCPC > router/eth0 (Rqst): (t1: 14400, t2: 25200)
01:17:47 DHCPC > router/eth0 (Rqst): Got DHCP lease for 172.16.224.102
01:17:47 DHCPC > router/eth0 (Rqst): Configuring IP interface
2001-01-01T01:17:48 : LOGINFO      : Link down on interface eth0.
2001-01-01T01:17:48 : LOGINFO      : Link up on interface eth0.
```

DHCP-server configuration tasks

To configure the SmartNode as DHCP-server perform the steps mentioned below.

- Configure DHCP-server profiles
- Use DHCP-server profiles and enable the DHCP-server (and to clear lease database) (see [page 252](#))
- Check DHCP-server configuration and status (see [page 253](#))
- Get debug output from the DHCP-server (see [page 253](#))

Configure DHCP-server profiles

The DHCP-server profiles hold the configuration information for the DHCP-server. The DHCP-server is capable of serving up to 8 subnets. Each subnet requires its own DHCP-server profile. The IP address/mask configuration of the IP interface implicitly links an IP interface to a subnet and hence to a DHCP-server profile.

Note A profile can only be modified if it is not assigned to the DHCP-server yet or if the DHCP-server is disabled. Use the command **no dhcp-server** to disable the DHCP-server (see below).

This procedure describes how to configure a DHCP-server profile

Mode: Configure

Step	Command	Purpose
1	node(cfg)#profile dhcp-server <i>name</i>	Enter DHCP-server profile mode
2	node(pf-dhcps)[name]#network <i>ip-address ip-mask</i>	Defines the IP address range for which this profile is responsible IP address: basic DHCP information ('your (client) IP address') IP mask: DHCP Option 1
3	node(pf-dhcps)[name]#[no] include <i>ip-address-from ip-address-to</i>	Defines up to 4 contiguous IP address ranges the server may use in the subnet defined in 2 (incremental command)
4	node(pf-dhcps)[name]#[no] default-router <i>default-router-ip-address</i>	Defines up to 2 default routers (default gateways) (incremental command) DHCP Option 3
5	node(pf-dhcps)[name]#lease infinite or node(pf-dhcps)[name]#lease <i>time days hours minutes</i>	Defines the time a lease is valid DHCP Option 51
6 (optional)	node(pf-dhcps)[name]#[no] domain-name <i>domain-name</i>	A PC DHCP client may use this domain name to complete host names to fully qualified domain names. DHCP Option 15
7 (optional)	node(pf-dhcps)[name]#[no] domain-name-server <i>domain-name-server-ip-address</i>	Defines up to 2 domain name servers (DNS) to be used by the client (incremental command) DHCP Option 6
8 (optional)	node(pf-dhcps)[name]#[no] netbios-name-server <i>netbios-name-server-ip-address</i>	Typical installation use <i>h-node</i> for hybrid. Refer to the Windows administration manuals for details about NetBIOS options. DHCP Option 44
9 (optional)	node(pf-dhcps)[name]#[no] netbios-node-type <i>b-node h-node m-node p-node</i>	Defines the NetBIOS node type (b: ???, h: hybrid, m: ???, p: ???) DHCP Option 46
10 (optional)	node(pf-dhcps)[name]#[no] bootfile <i>boot-file-name</i>	Defines the bootfile the client shall use when starting. Usually this is used in conjunction with the next-server command. Basic DHCP information ('Boot file name')

Step	Command	Purpose
11 (optional)	node(pf-dhcps)[name]#[no] next-server <i>next-server-ip-address</i>	Defines the address of the next server in the boot process. This could be a server different from the DHCP-server which provides configuration files for the clients to be downloaded. Basic DHCP information ('Next server IP address')

Example: Define a DHCP-server profile

This example shows how to configure a standard DHCP-server profile for a LAN with a private IP address range.

```
SN(cfg)#profile dhcp-server LAN
SN(pf-dhcps)[lan]#network 192.168.1.0 255.255.255.0
SN(pf-dhcps)[lan]#include 192.168.1.32 192.168.1.63
SN(pf-dhcps)[lan]#lease 2 days
SN(pf-dhcps)[lan]#default-router 192.168.1.1
SN(pf-dhcps)[lan]#domain-name-server 80.254.161.125
SN(pf-dhcps)[lan]#domain-name-server 80.254.161.126
```

Use DHCP-server profiles and enable the DHCP-server

If you have specified at least one profile, you can assign it to the DHCP-server and start the DHCP-server. This procedure describes how to assign one or more DHCP-server profiles and enable the DHCP-server

Mode: Context IP

Step	Command	Purpose
1	node(ctx-ip)[router]#[no] dhcp-server use name	Tell the DHCP-server (not) to use DHCP-server profile <i>name</i>
2	node(ctx-ip)[router]#[no] dhcp-server	Enables/disables DHCP-server
3	node(ctx-ip)[router]#dhcp-server clear-lease { all <i>ip-address</i> }	Removes all or a specific lease from the server's database, which in turn marks the IP address(es) as available again.

Example: Start the DHCP-server

This example shows how to assign a profile to the DHCP-server and to start the DHCP-server.

```
SN(ctx-ip)[router]#dhcp-server use LAN
SN(ctx-ip)[router]#dhcp-server
```

Check DHCP-server configuration and status

This procedure describes how to check the configuration and current status of the DHCP-server

Mode: Any

Step	Command	Purpose
1	node(cfg) #show dhcp-server	Displays configuration and status information

Example:

```
SN(ctx-ip)[router]#show dhcp-server
The DHCP server is running

Profiles

LAN (active)
Network           : 192.168.1.0 255.255.255.0
Include           : 192.168.1.32 - 192.168.1.63
Lease Time        : 2 days
Default Router    : 192.168.1.1
Domain Name Server : 80.254.161.125
                  : 80.254.161.126

Bound leases

192.168.1.32 (Dufour)
Address   : ethernet:00.10.A4.7C.7A.F8
Client Id : 01.00.10.A4.7C.7A.F8
Expires   : 2002-12-06T21:18:04
```

Get debug output from the DHCP-server

This procedure describes how to enable/disable the DHCP-server debug monitor

Mode: Any

Step	Command	Purpose
1	node(cfg) #[no] debug dhcp-server	Enables/disables the debug monitor of the DHCP-server

Example: Enable DHCP debug monitor

This example shows how to enable the DHCP-server debug monitor. The debug output shows an activation of the DHCP-server, a DHCP-client requesting a lease, and a DHCP-client releasing a lease.

```
SN(ctx-ip)[router]#debug dhcp-server

21:40:29  DHCPS > New network 'LAN' created

21:41:29  DHCPS > Discover from ethernet:00.10.A4.7C.7A.F8, client
id:01.00.10.A4.7C.7A.F8 via 192.168.1.1
21:41:29  DHCPS > Offering this hosts existing lease 192.168.1.32
21:41:29  DHCPS > Sending DHCP OFFER to 192.168.1.32 via 255.255.255.255 (68)
21:41:29  DHCPS > Deferring save of lease database
21:41:29  DHCPS > Last saved at 2002-12-04T21:40:29, next at 2002-12-04T21:55:29
21:41:29  DHCPS > Request from ethernet:00.10.A4.7C.7A.F8, client
id:01.00.10.A4.7C.7A.F8 via 192.168.1.1
21:41:29  DHCPS > Offer 192.168.1.32 has been selected
21:41:29  DHCPS > Sending DHCP ACK to 192.168.1.32 via 255.255.255.255 (68)
21:41:29  DHCPS > Deferring save of lease database
21:41:29  DHCPS > Last saved at 2002-12-04T21:40:29, next at 2002-12-04T21:55:29

21:44:37  DHCPS > Release from ethernet:00.10.A4.7C.7A.F8, client
id:01.00.10.A4.7C.7A.F8 via 192.168.1.1
21:44:37  DHCPS > Lease 192.168.1.32 released
21:44:37  DHCPS > Deferring save of lease database
21:44:37  DHCPS > Last saved at 2002-12-04T21:40:29, next at 2002-12-04T21:55:29
```


Chapter 22 **DNS configuration**

Chapter contents

Introduction	256
DNS configuration task list	256
Enabling the DNS resolver	256
Enabling the DNS relay	257

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 SmartNode uses a DNS resolver component to convert the DNS names into the numeric address.

The SmartNode 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 SmartNode, which checks to see if the DNS name is in its DNS resolver cache. If it finds the name in cache, the SmartNode 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 SmartNode 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 SmartNode 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 SmartNode DNS resolver, 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 SmartNode will query each server in turn until a response is received. DNS servers are configured as follows:

Mode: Configure

Step	Command	Purpose
1	node(cfg)#dns domain-name server <i>server-ip-address</i>	Add an IP address of a DNS server to be used resolving DNS names
2		Repeat step 1 for each additional DNS server you want to add
2	node(cfg)#dns-client cache <i>number-of-entries</i>	Optional. Defines the maximum number of DNS answers stored within the cache (default is 30)

Example: Configuring DNS servers

The following example shows how to add DNS servers to the SmartNode DNS resolver and increase the size of the DNS cache to 100 entries.

```
SN>enable
SN#configure
SN(cfg)#dns-client server 62.2.32.5
SN(cfg)#dns-client server 62.2.100.45
SN(cfg)#dns-client cache 100
```

You can test the DNS server configuration using the **dns-lookup** command as follows:

Example: Testing DNS server configuration

```
SN(cfg)#dns-lookup www.patton.com
Name:      www.patton.com
Address:   209.49.110.5
```

Enabling the DNS relay

The DNS resolver must be configured before you can use the DNS relay feature (see section “[Enabling the DNS resolver](#)” on page 256 to enable the DNS resolver, if you have not already done so).

Do the following to enable the DNS relay feature:

Mode: Configure

Step	Command	Purpose
1	node(cfg)#dns-relay	Enables DNS relay feature

Example: Enabling DNS relay

The following example shows how to enable the DNS relay feature.

```
SN>enable
SN#configure
SN(cfg)#dns-relay
```


Chapter 23 **DynDNS configuration**

Chapter contents

Introduction	260
DynDNS configuration task list	260
Creating a DynDNS account	260
Configuring the DNS resolver	260
Configuring basic DynDNS settings	261
Configuring advanced DynDNS settings (optional)	261
Defining a mail exchanger for your hostname	261
Troubleshooting	262

Introduction

SmartNodes are often used in applications where the addresses of their IP interfaces are not assigned statically (i.e. permanently) but instead are configured dynamically. In these applications, the IP address is assigned dynamically using protocols like DHCP or PPP. The problem with dynamically assigning addresses is that when the IP address changes, remote devices can no longer contact the SmartNode because they do not know what the new address is.

Dynamic DNS (DynDNS) addresses this problem by registering a permanent hostname for your SmartNode. DynDNS then directs traffic sent to the registered host name on to the SmartNode's ISP-assigned dynamic IP address, enabling the SmartNode to be accessed from the Internet without knowing its current dynamic IP address.

The DNS server used for registration is operated by Dynamic Network Services, Inc. You can find detailed information about the company and the services it offers on the webpage www.dyndns.org. The company offers different levels of service. The basic services are offered free of charge, while the more advanced services are chargeable.

The SmartNode supports the following DynDNS services:

- Dynamic DNS
- Static DNS
- Custom DNS

DynDNS configuration task list

This section describes configuring the DynDNS service. All possible configurations, which are involved in a specific configuration topic are described in the respective configuration task. To get a minimal working configuration of the DynDNS client, you must execute all the configuration tasks of the list below, except the tasks explicitly marked as optional.

- Creating a DynDNS account
- Configuring the DNS resolver
- Configuring basic DynDNS settings
- Configuring advanced DynDNS settings (optional)

Creating a DynDNS account

Before using the DynDNS service, you must create a DynDNS account on the DynDNS server and add a hostname to your account, which can be updated by the SmartNode. Go to the DynDNS website at www.dyndns.org and follow the instructions on the webpage to create the account and add a hostname.

Configuring the DNS resolver

The DynDNS client requires that the SmartNode's DNS resolver be enabled. You can find additional information about how to configure the DNS resolver in chapter 22, “[DNS configuration](#)” on page 255.

Configuring basic DynDNS settings

The following procedure describes the steps necessary to enable the DynDNS feature.

Mode: DynDNS

Step	Command	Purpose
1	node(dyndns)#authentication <i>user pass-word</i>	Defines the authentication credentials of your DynDNS account
2	node(dyndns)#service {dynamic static custom}	Defines the DynDNS service to use
3	node(dyndns)#hostname <i>name</i>	Defines the hostname that will be assigned to the SmartNode
4	node(dyndns)#observe <i>ip-interface-name</i>	Defines the IP interface to observe for IP address changes. When the IP address on this interface changes, the hostname to IP address mapping on the DynDNS server will be updated

Example: Configuring DynDNS

The following example shows the necessary steps required for a basic working configuration of the DynDNS client.

```
SN>enable
SN#configure
SN(cfg)#context ip
SN(ctx-ip)[router]#dyndns
SN(dyndns)#authentication Bob 245gf46te
SN(dyndns)#service dynamic
SN(dyndns)#hostname myhostname.dyndns.org
SN(dyndns)#observe eth1
```

Configuring advanced DynDNS settings (optional)

Defining a mail exchanger for your hostname

If required, you can define a mail exchanger or a backup mail exchanger for your hostname on the DynDNS server.

Mode: DynDNS

Step	Command	Purpose
1	node(dyndns)# mail-exchanger <i>hostname</i> [backup-mx]	Defines the host, which is the mail exchanger for your hostname. If the backup-mx parameter is specified, the mail-exchanger will be registered as backup mail exchanger only

Example: Defining a mail exchanger

The following example shows how to define a mail exchanger named *mail.mycompany.com*, which should be used as the primary mail-exchanger for the registered DynDNS hostname.

```
SN>enable
SN#configure
SN(cfg)#context ip
SN(ctx-ip)[router]#dyndns
SN(dyndns)#mail-exchanger mail.mycompany.com
```

Troubleshooting

The DynDNS component provides several commands to analyze and solve DynDNS problems. You can retrieve basic DynDNS status information as follows:

Mode: DynDNS

Step	Command	Purpose
1	node(dyndns)#show dyndns	Display basic DynDNS status information

Example: Displaying DynDNS status information

The following example displays status information of a properly configured and working DynDNS client.

```
SN(dyndns)#show dyndns
    Current state: Idle
    Last registered address: 243.232.39.64
    Hostname: test.dyndns.org
```

You can also monitor current activities of the DynDNS client. This includes ongoing DNS queries for DynDNS servers, verification of the currently registered IP address and updating the registration on the DynDNS server. The debug monitor can be enabled as follows;

Mode: Configure

Step	Command	Purpose
1	node(cfg)#debug dyndns	Enable the DynDNS debug monitor

Example: Displaying DynDNS status information

The following example shows how to enable the debug monitor and the output of the monitor when the IP address on the DynDNS server can be updated successfully.

```
SN(dyndns)#debug dyndns
    16:20:43  DYNDNS> Resolving 'checkip.dyndns.org'...
    16:20:43  DYNDNS> Resolved 'checkip.dyndns.org'.
    16:20:43  DYNDNS> Retrieving current IP address...
    16:20:43  DYNDNS> Sending request...
    16:20:43  DYNDNS> Current IP address (57.32.59.64) does not match last
registered one
                                           (43.23.44.2). DNS update is required.
    16:20:43  DYNDNS> Resolving 'update.dyndns.org'...
```



```

16:20:43 DYNDNS> Resolved 'update.dyndns.org'.
16:20:43 DYNDNS> Updating DNS...
16:20:43 DYNDNS> Sending request...
16:20:44 DYNDNS> DNS updated sucessfully
16:20:44 DYNDNS> Registered IP address is (57.32.59.64).

```

If required, you can force the DynDNS component to re-register the current IP address on the DynDNS server—even if the dynamic IP address has not changed—using the following command (this command could also be useful for observing the update process in the debug monitor).



Possible blocking — Do not use this command too often, because the DynDNS server will block your hostname, if you trigger too many unnecessary updates of your IP address.

You can also force the DynDNS client to resume normal operation, if the state of the DynDNS client is shown as blocked and the problem which led to the blocked state has been solved. The DynDNS client will enter the blocked state if the DynDNS server reports an unrecoverable error during DNS updates that require user intervention. These are mainly configuration problems, such as invalid credentials or an invalid hostname.

Mode: DynDNS

Step	Command	Purpose
1	<code>node(dyndns)#dyndns reset</code>	Forces a re-registration of the current IP address on the DynDNS server, even if an update is not necessary

Chapter 24 **PPP configuration**

Chapter contents

Introduction	266
PPP configuration task list	267
Creating an IP interface for PPP	267
Creating a PPP subscriber	269
Configuring a PPPoE session	270
Configuring a serial port for PPP	272
Creating a PPP profile	273
Displaying PPP configuration information	274
Debugging PPP	275
Sample configurations	279
PPP over Ethernet (PPPoE)	279
Without authentication, encapsulation multi, with NAPT	279
With authentication, encapsulation PPPoE	279
PPP over serial link	280
Without authentication, numbered interface	280
With authentication, unnumbered interface	280

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. SmartWare offers PPP over the following link layers:

- PPP over Ethernet (PPPoE)
- PPP over serial link, i.e. V.35/X.21, HDLC (only available with the SmartNode 2300)

Figure 48 shows the elements involved in the configuration of PPP. The elements required to configure PPP over Ethernet are located in the upper left corner of the figure. The elements for PPP over serial link are in the lower left corner. For PPP over ISDN, the elements are in the middle and the lower right corner.

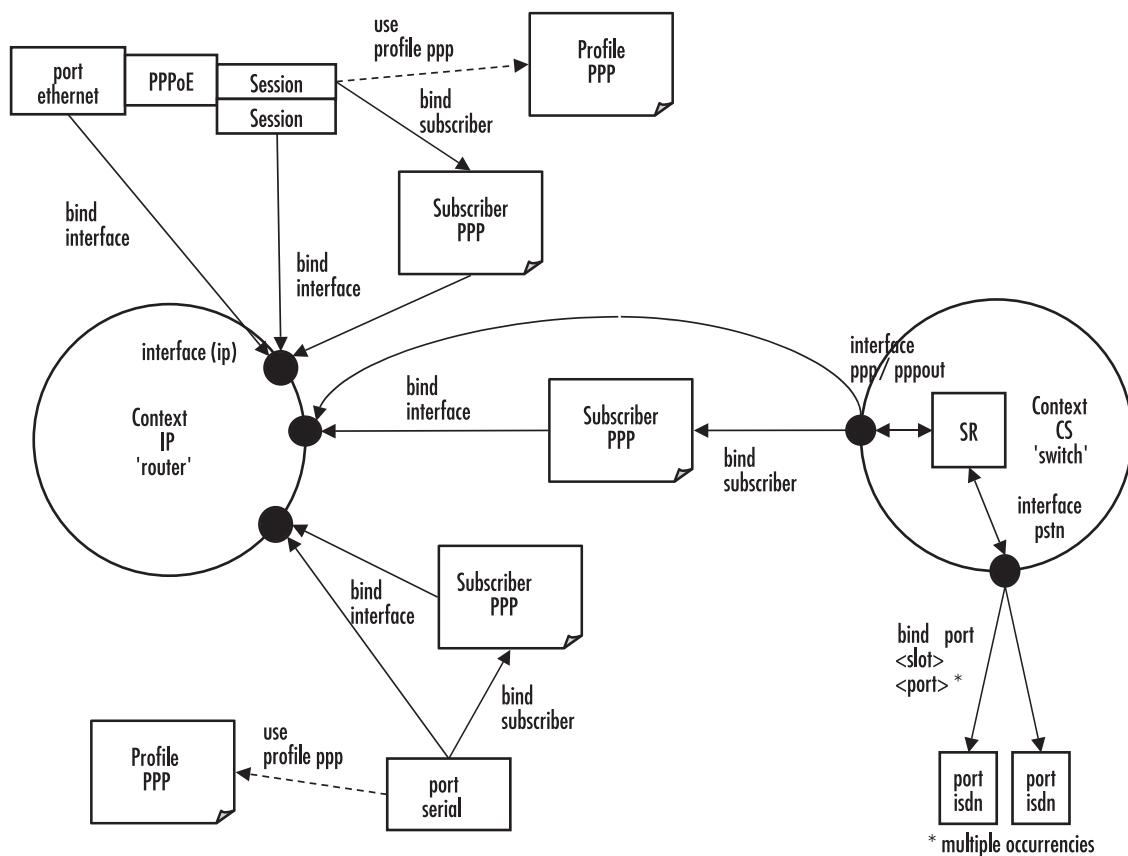


Figure 48. PPP configuration overview

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 a subscriber profile if either PPP peer requires authentication.

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
- Creating a PPP subscriber (for authentication) (see [page 269](#))
- Configuring a PPPoE session (see [page 270](#))
- Configuring a serial port for PPP (see [page 272](#))
- Creating a PPP interface within the CS context (not currently available)
- Creating a PSTN interface for PPP dial-in/dial-out (not currently available)
- Creating a PPP profile (see [page 273](#))
- Displaying PPP configuration information (see [page 274](#))
- Debugging PPP (see [page 275](#))

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 PPP service provider only offers a single IP address and not an IP subnet, or if the IP addresses on the LAN shall be private and hidden behind a public IP address (see 12, “[NAT/NAPT configuration](#)” on page 133 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 name</code>	Creates the new interface <i>name</i> , which represents an IP interface.
2	<code>node(if-ip)[name]#point-to-point</code>	Only defines what route is entered into the IP routing table: <ul style="list-style-type: none"> • point-to-point: A route to the IP address of the PPP interface (assigned by the PPP peer) is entered into the routing table. • no point-to-point: A route to the subnet defined by IP address of the PPP interface (assigned by the PPP peer) is entered into the routing table. The class of the IP address determines the size of the subset. Recommendation: Use 'point-to-point' and specify a default route.

Step	Command	Purpose
3	<p><code>node(if-ip)[name]#ipaddress unnumbered</code></p> <p>or</p> <p><code>node(if-ip)[name]#ipaddress dhcp</code></p> <p>or</p> <p><code>node(if-ip)[name]# ipaddress ip-address netmask</code></p>	<p>The PPP remote peer offers an IP address for the IP interface. The IP interface adopts this IP address</p> <p>Once PPP has established an IP connection, the IP interface can use DHCP to acquire an IP address. It sends a DHCP Discover message (which is an IP broadcast) to the IP network to which PPP has established connection. If no DHCP Server is present, the IP interface does not adopt the IP address offered by the PPP remote peer but leaves the IP address undefined.</p> <p>The IP interface requests from the PPP remote peer to use the IP address <i>ip-address</i>. PPP repeatedly tries to set-up a connection until the remote peer accepts this IP address. It does not accept any other IP address offered by the PPP remote peer. The parameter <i>netmask</i> specifies the size of the subnet in case 'no point-to-point' is configured</p>
4 (optional)	<p><code>node(if-ip)[name]# [no] tcp adjust-mss { rx tx } { mtu mss }</code></p>	<p>Limits to the MSS (Maximum Segment Size) in TCP SYN packets to <i>mss</i> or to MTU (Maximum Transmit Unit) - 40 Bytes, if 'mtu' is used. 'rx' applies to packets which arrive inbound at this IP interface, 'tx' to packets which leave outbound of this IP interface.</p> <p>PPP over Ethernet connections impose an overhead of 8 Bytes (PPP: 2 Bytes, PPPoE: 6 Bytes). Some Ethernets do not allow payloads larger than the 1500 Bytes which the standard defines, so IP packets must not contain more than 1492 bytes when transmitted over such connections. Reducing the MTU/MRU to 1492 Bytes does not always solve the problem because many sources do not allow fragmentation of the IP packets they send (they set the 'Don't fragment'). However, these sources limit the size of the IP packets according to the MSS which their peers announce in the TCP SYN packets.</p> <p>It is recommended to use 'mtu' inbound and outbound.</p>

Step	Command	Purpose
5 (optional)	<code>node(if-ip)[name]#use profile napt name</code>	Assigns the NAPT profile <i>name</i> to applied to this IP interface. See 12, “ NAT/NAPT configuration ” on page 133 to learn how to create a NAPT profile.

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.

```
SN(cfg)#context ip router
SN(ctx-ip)[router]#interface ppp_interface
SN(if-ip)[ppp_int~]#point-to-point
SN(if-ip)[ppp_int~]#ipaddress unnumbered
SN(if-ip)[ppp_int~]#tcp adjust-mss rx mtu
SN(if-ip)[ppp_int~]#tcp adjust-mss tx mtu
```

Creating a PPP subscriber

One or more PPP subscriber shall be configured if either PPP peer requires authentication. This procedure describes how to create a PPP subscriber

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg) # subscriber ppp name</code>	Creates the new subscriber <i>name</i> , which contains the authentication settings.
2	<code>node(subscr)[name]# dial {in out}</code>	Defines the direction of the connection establishment with PPP over ISDN. This information allows to use different subscribers for incoming and outgoing calls. With the other two link layers, set the direction as follows: <ul style="list-style-type: none"> • PPP over Ethernet: ‘dial out’ • PPP over Serial: ‘dial in’
3	<code>node(subscr)[name]# [no] authentication { (chap pap) {chap pap} }</code>	Defines the authentication protocol to be used, PAP and/or CHAP

Step	Command	Purpose
4 (optional)	node(subscr)[name]# [no] identification {outbound inbound} user [password password]	<p>Sets the credentials to be provided during the authentication procedure: the user name <i>user</i> and the password <i>password</i>.</p> <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 • - PPP over Serial: 'inbound only'
5	node(subscr)[name]# [no] bind interface interface [router]	Binds the subscriber 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 267.

Example: Create a PPP subscriber

The procedure below creates a PPP subscriber for a PAP authentication with some Internet Service Provider.

```
SN(cfg)#subscriber ppp joe_example
SN(subscr)[joe_exa~]#dial out
SN(subscr)[joe_exa~]#authentication pap
SN(subscr)[joe_exa~]#identification outbound joeexample@isp.com password blue4you
SN(subscr)[joe_exa~]#bind interface ppp_interface router
```

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	node(cfg) #port ethernet <i>slot port</i>	Enters Ethernet port configuration mode for the interface on <i>slot</i> and <i>port</i>
2	node (prt-eth)[slot/port]# encapsulation {ip pppoe multi}	Defines the payload type(s) to be used on the Ethernet: <ul style="list-style-type: none"> • 'ip': IP traffic only (not used for PPP) • 'pppoe': PPPoE sessions only • 'multi': both IP traffic and PPPoE sessions
3	node (prt-eth)[slot/port]# [no] bind interface <i>name</i> [<i>router</i>]	Binds the Ethernet port to the IP interface to be used for the direct IP traffic (only required if encapsulation 'ip' or 'multi' is selected)
4	node(prt-eth)[slot/port]#[no] shutdown	Enables the ethernet port
5	node(prt-eth)[slot/port]#pppoe	Enters PPPoE mode
6	node(pppoe)[slot/port]#session <i>name</i>	Creates PPPoE session with the name <i>name</i>
7	node(pppoe)[slot/port]# [no] bind interface <i>name</i> [<i>router</i>] or node (pppoe)[slot/port]# [no] bind subscriber <i>name</i>	Binds the PPPoE session directly to the IP interface <i>name</i> in case no authentication is required Binds the PPPoE session to the PPP subscriber <i>name</i> in case authentication is required
8 (optional)	node (pppoe)[slot/port]# [no] use profile ppp <i>name</i>	Assigns a PPP profile other than the default profile to this PPPoE session
9 (optional)	node(session)[name]#service <i>Service-Name</i>	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)
10 (optional)	node(session)[name]#access-concentrator <i>AC-Name</i>	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
11	node(session)[name]#[no] shutdown	Initiates the establishment of the PPPoE session and the PPP connection

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 subscriber profile above.

```
SN(cfg)#port ethernet 0 0
SN(prt-eth)[0/0]#encapsulation pppoe
SN(prt-eth)[0/0]#no shutdown
SN(prt-eth)[0/0]#pppoe
SN(pppoe)[0/0]#session green
SN(session)[green]#bind subscriber joe_example
SN(session)[green]#no shutdown
```

Configuring a serial port for PPP

PPP can run over serial ports, e.g. the X.21/V.35 port on the SmartNode 2300. This procedure describes how to configure a serial port for PPP

Mode: Configure

Step	Command	Purpose
1	node(cfg)#port serial <i>slot port</i>	Enters the configuration mode for the serial port on <i>slot</i> and <i>port</i>
2	node(prt-ser)[slot/port]# [no] encapsulation { framerelay ppp }	Sets the encapsulation to 'ppp'
3	node (prt-ser)[slot/port]# [no] bind interface <i>name</i> [router] or node (prt-ser)[slot/port]# [no] bind subscriber <i>name</i> or node (prt-ser)[slot/port]# [no] bind subscriber authentication { chap pap { chap pap } }	Binds the serial port directly to the IP interface <i>name</i> in case no authentication is required Binds the serial port to the PPP subscriber <i>name</i> in case authentication is required Only the credentials provided at the establishment of the PPP session select the PPP subscriber. This allows to bind the serial port to the set of all PPP subscribers.
4 (optional)	node (prt-ser)[slot/port]# [no] use profile ppp name	Assigns a PPP profile other than the default profile to this serial port
5	node(prt-ser)[slot/port]#[no] shutdown	Enables the serial port and initiates the establishment of the PPP connection

Example: Configure a serial port for PPP

The procedure below configures the serial port for a leased-line connection to an Internet Service Provider using the credentials specified in the subscriber profile above.

```
SN(cfg)#port serial 0 0
SN(prt-ser)[0/0]#encapsulation ppp
SN(prt-ser)[0/0]#bind subscriber joe_example
```

```
SN(prt-ser)[0/0]#no shutdown
```

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 no profile can 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.

This procedure describes how to create a PPP profile or to modify the default PPP profile

Mode: Configure

Step	Command	Purpose
1	node(cfg) #[no] profile ppp { name default }	Creates the new PPP profile <i>name</i> and enters the PPP profile configuration. The profile 'default' already exists.
2 (optional)	node(pf-ppp)[name]#mtu min <i>min</i> max <i>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)	node(pf-ppp)[name]#mru min <i>min</i> max <i>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)	node(pf-ppp)[name]#[no] van-jacobson {compression decompression} max-slots <i>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.

```
SN(cfg)#profile ppp PPPoE
SN(pf-ppp)[PPPoE]#mtu min 68 max 1492
```

```

SN(pf-ppp)[PPPoE]#mru min 68 max 1492
SN(pf-ppp)[PPPoE]#van-jacobson compression
SN(pf-ppp)[PPPoE]#port ethernet 0 0
SN(prt-eth)[0/0]#pppoe
SN(pppoe)[0/0]#session green
SN(session)[green]#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	<p>Gives the best overview of all PPP related configuration information. The following parts are of interest:</p> <ul style="list-style-type: none"> • profile ppp default • profile ppp <i>name</i> • interface <i>name</i> • subscriber ppp <i>name</i> • port ethernet <i>slot port</i> • session <i>name</i>
2	node(cfg) #show subscriber ppp [<i>name</i>]	Displays configuration information of the PPP subscriber <i>name</i> or of all PPP subscribers
3	node(pf-ppp)[<i>name</i>]#show profile ppp { <i>name</i> default }	Displays the PPP profile <i>name</i> or the default PPP profile

Example: Display PPP subscriber configuration information

```

SN(session)[green]#show subscriber ppp joe_example

Subscribers:
-----

Name:                joe_example
Direction:          dial-out
Authentication:     pap
Identification (inbound): (none)
Identification (outbound): patton/patton
Timeout for disconnect: no absolute timeout, no idle timeout
Max. sessions:      no limit
IP address:         (none)
Callback:           (none)
Binding:            interface ppp_interface router
Binding:            interface ppp_interface router

```

Example: Display a PPP profile

```
SN(pf-ppp)[PPPoE]#show profile ppp PPPoE
```

```
Profiles:
```

```
-----
```

```
Name:                                default
LCP Configure-Request:                interval 3000 ms, max 10
LCP Configure-Nak:                    max 5
LCP Terminate-Request:                interval 3000 ms, max 2
LCP Echo-Request:                     interval 10000 ms, max 3
MTU:                                   68 - 1920
MRU:                                   68 - 1920
Callback:                              both
CHAP:                                   allowed
PAP:                                    allowed
Authentication:                       interval 3000 ms, max 3
IPCP Configure-Request:                interval 3000 ms, max 10
IPCP Configure-Nak:                    max 5
IPCP Terminate-Request:                interval 3000 ms, max 2
Van-Jacobson Compression:              allowed, max-slots 31
Van-Jacobson Decompression:            allowed, max-slots 31
```

```
Name:                                PPPoE
LCP Configure-Request:                interval 3000 ms, max 10
LCP Configure-Nak:                    max 5
LCP Terminate-Request:                interval 3000 ms, max 2
LCP Echo-Request:                     interval 10000 ms, max 3
MTU:                                   68 - 1492
MRU:                                   68 - 1492
Callback:                              both
CHAP:                                   allowed
PAP:                                    allowed
Authentication:                       interval 3000 ms, max 3
IPCP Configure-Request:                interval 3000 ms, max 10
IPCP Configure-Nak:                    max 5
IPCP Terminate-Request:                interval 3000 ms, max 2
Van-Jacobson Compression:              allowed, max-slots 24
Van-Jacobson Decompression:            allowed, max-slots 31
Van-Jacobson Decompression:            allowed, max-slots 31
```

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	node(cfg) #show ppp links [level]	<p>Displays status and configuration information of the Link Control Protocol (LCP) and the authentication protocol(s) (PAP and/or CHAP). Check whether the states of the two protocols are 'Opened'.</p> <p>level specifies to level of details displayed (1..4, default is 1).</p>
2	node(cfg) #show ppp networks [level]	<p>Displays status and configuration information of the Network Control Protocol(s) (NCP), in particular the IP Control Protocol (IPCP). Check whether the states of this protocol is 'Opened'.</p> <p>Under 'Local configuration options', you find the IP address proposed by this SmartNode and under 'Local acknowledged options', the IP address assigned by the remote peer.</p> <p>level specifies to level of details displayed (1..4, default is 1).</p>
3	node(cfg) #show pppoe [name]	<p>Displays status, configuration information, and statistics of PPPoE in general and of the PPPoE session(s). Check whether state of the respective session is 'Opened'.</p> <p>level specifies to level of details displayed (1..4, default is 1).</p>
4	node(cfg) #show port interface name	<p>Displays status and configuration information of the IP interface at which a PPP connection terminates. Check whether state of the interface is 'OPENED'.</p> <p>Under 'Local IP Address', you find the IP address assigned to the IP interface. If it does not correspond to the IP address assigned by the PPP remote peer, check whether the 'ipaddress' of the IP interface is set to 'unnumbered'.</p>
5	node(cfg) #show port ethernet slot port or node(cfg) #show port serial slot port	<p>Displays status and configuration information of the Ethernet/serial port over which a PPP connection/PPPoE sessions runs. Check whether state of the port is 'OPENED' and whether the encapsulation is set to 'pppoe' or 'multi' (only for Ethernet ports).</p>
6	node(cfg) # [no] debug ppp [all ...]	Enables all or a particular PPP debug monitor.
7	node(cfg) # [no] debug pppoe [all ...]	Enables all or a particular PPPoE debug monitor.

Example: Display PPP link information

```

SN(cfg)#show ppp links 4

PPP Link Information:
=====
Link:
  ID:                0
  Name:              ethernet 0 0 0/pppoe/ppp_green
  Protocols:        LCP, PAP
LCP:
  ID:                0
  Name:              ethernet 0 0 0/pppoe/ppp_green
  State:             Opened
  Conf-Req send rate: 3000ms
  Max. Conf-Req:     10
  Term-Req send rate: 3000ms
  Max. Term-Req:     2
  Echo-Req send rate: 10000ms
  Max. Echo-Req:     3
  Local ID:          100000020390
  Remote ID:
  Local configured options:
    Magic Number = 0x00000000
    MRU = 1492 [68,1492]
    ACCM = 0xffffffff
  Local acknowledged options:
  Remote configured options:
    Magic Number = 0xb89d9e6b
    MRU = 1492 [68,1492]
    ACCM = 0xffffffff
    Authentication Protocol = { PAP }
  Remote acknowledged options:
    MRU = 1492 [68,1492]
    Magic Number = 0xb89d9e6b
    Authentication Protocol = { PAP }
  Remote denied options:
  Remote rejected options:
PAP:
  ID:                0
  Name:              ethernet 0 0 0/pppoe/ppp_green
  State:             Opened
  Direction:         supplying
  Local authentication:
    ID:              patton
    Password:        patton
    Success:
  Remote authentication:
    ID:
    Password:
    Success:         Greetings!!
  Auth-Req send rate: 3000ms
  Max. Auth-Req:     3

```

Example: Display PPP network protocol information

```
SN(session)[green]#show ppp networks 4

PPP Network Information:
=====
Network:
  ID:                0
  Name:              ethernet 0 0 0/pppoe/ppp_green/net
  State:             up
IPCP:
  ID:                0
  Name:              ethernet 0 0 0/pppoe/ppp_green/net
  State:             Opened
  Conf-Req send rate: 3000ms
  Max. unanswered Conf-Req: 10
  Local configured options:
    IP Address = 172.16.40.98
    IP Compression Protocol = VJC (Max-Slot-Id=31, Comp-Slot-Id=1)
  Local acknowledged options:
    IP Address = 10.10.10.2
    IP Compression Protocol = VJC (Max-Slot-Id=31, Comp-Slot-Id=1)
  Remote configured options:
    IP Address = 0.0.0.0
    IP Compression Protocol = VJC (Max-Slot-Id=24, Comp-Slot-Id=1)
  Remote acknowledged options:
    IP Address = 10.10.10.1
    IP Compression Protocol = VJC (Max-Slot-Id=15, Comp-Slot-Id=1)
  Remote denied options:
  Remote rejected options:
```

Example: Display PPPoE information

```
SN(session)[green]#show pppoe 4

PPPoE Information:
=====
Instance:
  ID:                0
  Name:              ethernet 0 0 0/pppoe
  Initiation Send Interval 3000 ms
  Request Send Interval 1000 ms
  Max. Initiations      20
  Max. Requests         3
  Received Octets       7247
  Received Packets     181
  Received Discards     0
  Received Errors       2
  Received Unknown Protos 0
  Transmitted Octets    2952
  Transmitted Packets  152
  Transmitted Discards  1
  Transmitted Errors    0
  Session:
    ID:                1
    Name:              green
```



```

Service:
Access-Concentrator:
State:                Opened
Sent Initiations:     1
Sent Requests:       1
Peer Session-ID:     3786
Peer MAC-Address:    00:01:02:B8:4E:E4

```

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 unnumbered
        point-to-point
        tcp adjust-mss rx mtu
        tcp adjust-mss tx mtu
        use profile napt WAN

context ip router
    route 0.0.0.0 0.0.0.0 ppp_interface 0

port ethernet 0 0
    encapsulation multi
    bind interface normal_ip_interface
    no shutdown

pppoe

    session green
        bind interface ppp_interface
        no shutdown

```

With authentication, encapsulation PPPoE

```

context ip router

    interface ppp_interface
        ipaddress unnumbered
        point-to-point
        tcp adjust-mss rx mtu
        tcp adjust-mss tx mtu

subscriber ppp joe_example
    dial out
    authentication pap

```

```
    identification outbound <user> password <password>
    bind interface ppp_interface router

port ethernet 0 0
  encapsulation pppoe
  no shutdown

  pppoe

    session green
      bind subscriber joe_example
      no shutdown
```

PPP over serial link

Without authentication, numbered interface

```
context ip router

  interface ppp_interface
    ipaddress 172.17.1.1 255.255.255.252
    point-to-point

port serial 0 0
  encapsulation ppp
  bind interface ppp_interface
  no shutdown
```

With authentication, unnumbered interface

```
context ip router

  interface ppp_interface
    ipaddress unnumbered
    point-to-point

subscriber ppp joe_example
  dial in
  authentication pap
  identification inbound <user> password <password>
  bind interface ppp_interface router

port serial 0 0
  encapsulation ppp
  bind interface ppp_interface
  no shutdown
```

Chapter 25 VPN configuration

Chapter contents

Introduction	282
Authentication	282
Encryption	282
Transport and tunnel modes	283
Key management	283
VPN configuration task list	283
Creating an IPsec transformation profile	283
Creating an IPsec policy profile	284
Creating/modifying an outgoing ACL profile for IPsec	286
Configuration of an IP interface and the IP router for IPsec	287
Displaying IPsec configuration information	287
Debugging IPsec	288
Sample configurations	289
IPsec tunnel, DES encryption	289
SmartNode configuration	289
Cisco router configuration	290
IPsec tunnel, AES encryption at 256 bit key length, AH authentication with HMAC-SHA1-96	290
SmartNode configuration	290
Cisco router configuration	291
IPsec tunnel, 3DES encryption at 192 bit key length, ESP authentication with HMAC-MD5-96	291
SmartNode configuration	291
Cisco router configuration	291

Introduction

This chapter describes how to configure the VPN connections between two SmartNodes or between a SmartNode and a third-party device.

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.

There are different technologies to implement a VPN. SmartWare applies the *internet protocol security* (IPsec) Architecture (see RFC 2401). The following sections describe the main building blocks of the IPsec architecture as implemented in SmartWare.

Authentication

Authentication verifies the integrity of data stream and ensures that it is not tampered with while in transit. It also provides confirmation about data stream origin.

Two authentication protocols are available:

- Authentication header (AH): protects the IP payload, the IP header, and the authentication header itself
- Encapsulating security payload (ESP): protects the IP payload and the ESP header and trailer, but not the IP header

Two algorithms perform the authentication:

- HMAC-MD5-96: is a combination of the *keyed-hashing for message authentication* (HMAC) and the *message digest version 5* (MD5) hash algorithm. It requires an authenticator of 128-bit length and calculates a hash of 96 bits over the packet to be protected (see RFC 2403).
- HMAC-SHA1-96: is a combination of the (HMAC) and the *secure hash algorithm version 1* (SHA1). It requires an authenticator of 160 bit length and calculates a hash of 96 bits over the packet to be protected (see RFC 2404).

Encryption

Encryption protects the data in transit from unauthorized access. Encapsulating security payload (ESP) is the protocol to transport encrypted IP packets over IP (see RFC 2406).

The following encryption algorithms are available:

	Key Length [Bit]	RFC
DES-CBC (Data Encryption Standard - Cipher Block Chaining)	56	2405
3DES-CBC (Triple Data Encryption Standard - Cipher Block Chaining)	128 or 192 ^a	1851
AES-CBC (Advanced Encryption Standard - Cipher Block Chaining)	128, 192, or 256	3268

a. The 3DES algorithm uses only 112 out of the 128 Bit or 168 out of the 192 Bit as key information. Cisco only supports 192 Bit keys with 3DES.

The single DES algorithm no longer offers adequate security because of its short key length (a minimum key length 100 bits is recommended). The AES algorithm is very efficient and allows the fastest encryption. AES with a key length of 128 bits is therefore the recommended algorithm.

Transport and tunnel modes

The mode determines the payload of the ESP packet and hence the application:

- Transport mode: Encapsulates only the payload of the original IP packet, but not its header, so the IPsec peers must be at the endpoints of the communications link.
- A secure connection between two hosts is the application of the transport mode.
- Tunnel mode: Encapsulates the payload and the header of the original IP packet. The IPsec peers can be (edge) routers that are not at the endpoints of the communications link.

A secure connection of the two (private) LANs, a 'tunnel', is the application of the tunnel mode.

Key management

The current implementation of IPsec in SmartWare works with pre-shared keys (also called *manual keying* or *manual IPsec*). Keys are manually generated, distributed, and stored as a hexa-decimal string in the startup-configuration of the SmartNode and its peer.

Note Depending on the processing hardware applied to *reverse engineering* a DES key, it can take from 3 hours to 3 days to break the key. Thus, for maximum security, DES keys must be manually updated regularly. AES- or 3DES-keys, because they are much more complex, take so much longer to break as to be practically infinite.

The Internet key exchange (IKE) protocol is not currently supported.

VPN configuration task list

To configure a VPN connection, perform the following tasks:

- Creating an IPsec transformation profile
- Creating an IPsec policy profile
- Creating/modifying an outgoing ACL profile for IPsec
- Configuration of an IP Interface and the IP router for IPsec
- Displaying IPsec configuration information
- Debugging IPsec

Creating an IPsec transformation profile

The IPsec transformation profile defines which authentication and/or encryption protocols, which authentication and/or encryption algorithms shall be applied.

Procedure: To create an IPsec transformation profile

Mode: Configure

mac-sha1-96 } Enables authentication and defines the authentication protocol and the hash algorithm

Step	Command	Purpose
1	node(cfg)#profile ipsec-transform <i>name</i>	Creates the IPsec transformation profile <i>name</i>
2 optional	node(pf-ipstr)[<i>name</i>]#esp-encryption { aes-cbc des-cbc 3des-cbc } [<i>key-length</i>]	Enables encryption and defines the encryption algorithm and the key length Supported key lengths see section “Encryption” on page 282
3 optional	node(pf-ipstr)[<i>name</i>]#{ ah-authentication esp-authentication } {hmac-md5-96 hmac-sha1-96 }	Enables authentication and defines the authentication protocol and the hash algorithm

Use **no** in front of the above commands to delete a profile or a configuration entry.

Example: Create an IPsec transformation profile

The following example defines a profile for AES-encryption at a key length of 128.

```
SN(cfg)#profile ipsec-transform AES_128
SN(pf-ipstr)[AES_128]#esp-encryption aes-cbc 128
```

Creating an IPsec policy profile

The IPsec policy profile supplies the keys for the encryption and/or the authenticators for the authentication, the *security parameters indexes* (SPIs), and IP address of the peer of the secured communication. Furthermore, the profile defines which IPsec transformation profile to apply and whether transport or tunnel mode shall be most effective.

The SPI identifies a secured communication channel. The IPsec component needs the SPI to select the suitable key or authenticator. Inbound and outbound channels can have the same SPI, but the channels in the same direction—inbound or outbound—must have unique SPIs. The SPI is not encrypted and can be monitored.

Procedure: To create an IPsec policy profile

Mode: Configure

Step	Command	Purpose
1	node(cfg)#profile ipsec-policy-manual <i>name</i>	Creates the IPsec policy profile name
2	node(pf-ipstr)[name]#use profile ipsec-transform <i>name</i>	Selects the IPsec transformation profile to be applied
3 optional	node(pf-ipstr)[name]#session-key { inbound outbound } { ah-aauthentication esp-aauthentication esp-encryption } <i>key</i>	<p>Sets a key for encryption or an authenticator for authentication, either for inbound or outbound direction. The key shall consist of hexadecimal digits (0..9, A..F); one digit holds 4 Bit of key information.</p> <p>The key setting must match definitions in the respective IPsec transformation profile. In particular, the length of the key or authenticator must match the implicit (see section “Authentication” on page 282 and “Encryption” on page 282) or explicit specification.</p> <p>Keys must be available for inbound and outbound directions. They can be different for the two directions. Make sure that the inbound key of one peer matches the outbound key of the other peer.</p>
4	node(pf-ipstr)[name]#spi { inbound outbound } { ah esp } <i>spi</i>	<p>Sets the SPI for encryption (esp) or authentication (ah), either for inbound or outbound direction. The SPI shall be a decimal figure in the range $1..2^{32}-1$.</p> <p>SPIs must be available for encryption and/or authentication as specified in the respective IPsec transformation profile.</p> <p>SPIs must be available for inbound and outbound directions. They can be identical for the two directions but must be unique in one direction. Make sure that the inbound SPI of one peer matches the outbound SPI of the other peer.</p>
5	node(pf-ipstr)[name]#peer <i>ip-address</i>	<p>Sets the IP address of the peer</p> <p>Note The peers of the secured communication must have static IP address. DNS resolution is not available yet.</p>
6	node(pf-ipstr)[name]#mode { tunnel transport }	Selects tunnel or transport mode

Use **no** in front of the above commands to delete a profile or a configuration entry.

Example: Create an IPsec policy profile

The following example defines a profile for AES-encryption at a key length of 128.

```
SN(cfg)#profile ipsec-policy-manual ToBerne
SN(pf-ipsma)[ToBerne]#use profile ipsec-transform AES_128
SN(pf-ipsma)[ToBerne]#session-key inbound esp-encryption
1234567890ABCDEF1234567890ABCDEF
SN(pf-ipsma)[ToBerne]#session-key outbound esp-encryption
FEDCBA0987654321FEDCBA0987654321
SN(pf-ipsma)[ToBerne]#spi inbound esp 1111
SN(pf-ipsma)[ToBerne]#spi outbound esp 2222
SN(pf-ipsma)[ToBerne]#peer 200.200.200.1
SN(pf-ipsma)[ToBerne]#mode tunnel
```

Creating/modifying an outgoing ACL profile for IPsec

An access control list (ACL) profile in the outgoing direction selects which outgoing traffic to encrypt and/or authenticate, and which IPsec policy profile to use. IPsec does not require an incoming ACL.

Note Outgoing and incoming IPsec traffic passes an ACL (if available) twice, once before and once after encryption/authentication. So the respective ACLs must permit the encrypted/authenticated and the plain traffic.

For detailed information on how to set-up ACL rules, see chapter 18, “Access control list configuration” on page 203.

Procedure: To create/modify an outgoing ACL profile for IPsec

Mode: Configure

Step	Command	Purpose
1	node(cfg)#profile acl name	Creates or enters the ACL profile name
2	node(pf-ipstr)[name]#permit ... [ipsec-policy name]	The expression ‘ipsec-policy name’ appended to a permit ACL rule activates the IPsec policy profile <i>name</i> to encrypt/authenticate the traffic identified by this rule.

Note New entries are appended at the end of an ACL. Since the position in the list is relevant, you might need to delete the ACL and rewrite it completely.

Example: Create/modify an ACL profile for IPsec

The following example configures an outgoing ACL profile that interconnects the two private networks 192.168.1/24 and 172.16/16.

```
SN(cfg)#profile acl VPN_Out
SN(pf-acl)[VPN_Out]#permit ip 192.168.1.0 0.0.0.255 172.16.0.0 0.0.255.255 ipsec-
policy ToBerne
SN(pf-acl)[VPN_Out]#permit ip any any
```


Configuration of an IP interface and the IP router for IPsec

The IP interface that provides connectivity to the IPsec peer, must now activate the outgoing ACL profile configured in the previous section. Furthermore, the IP router must have a route for the remote network that points to the respective IP interface.

Procedure: To activate the outgoing ACL profile and to establish the necessary route

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg)#context ip router</code>	Enter IP context
2	<code>node(ctx-ip)[router]#interface if-name</code>	Create/enter the IP interface <i>if-name</i>
3	<code>node(if-ip)[if-name]# use profile acl name out</code>	Activate the outgoing ACL profile <i>name</i>
4	<code>node(if-ip)[if-name]#context ip router</code>	Enter IP context
5 optional	<code>node(ctx-ip)[router]#route remote-network-address remote-network-mask if-name 0</code>	Creates a route for the remote network that points the above IP interface <i>if-name</i> You can omit this setting if the default route already points to this IP interface or to a next hub reachable via this IP interface, and if there is no other route. Make also sure that the IP router knows how to reach the peer of the secured communication. Usually, a default route does this job.

Example: Activate outgoing ACL and establish route

The following example configures an outgoing ACL profile that interconnects the two private networks 192.168.1/24 and 172.16/16.

```
SN(cfg)#context ip router
SN(ctx-ip)[router]#interface WAN
SN(if-ip)[WAN]#use profile acl VPN_Out out
SN(if-ip)[WAN]#context ip router
SN(ctx-ip)[router]#route 172.16.0.0 255.255.0.0 WAN 0
```

Displaying IPsec configuration information

This section shows how to display and verify the IPsec configuration information.

Procedure: To display IPsec configuration information

Mode: Configure

Step	Command	Purpose
1 optional	<code>node(cfg)#show profile ipsec-trans-form</code>	Displays all IPsec transformation profiles
2 optional	<code>node(cfg)#show profile ipsec-policy-manual</code>	Displays all IPsec policy profiles

Example: Display IPsec transformation profiles

```
SN(cfg)#show profile ipsec-transform

IPSEC transform profiles:

Name: AES_128
  ESP Encryption: AES-CBC, Key length: 128
```

Example: Display IPsec policy profiles

```
SN(cfg)#show profile ipsec-policy-manual

Manually keyed IPsec policy profiles:

Name: ToBerne, Peer: 200.200.200.1, Mode: tunnel, transform-profile: AES_128
  ESP SPI Inbound: 1111, Outbound: 2222
  ESP Encryption Key Inbound: 1234567890ABCDEF1234567890ABCDEF
  ESP Encryption Key Outbound: FEDCBA0987654321FEDCBA0987654321
```

Debugging IPsec

A debug monitor and an additional show command are at your disposal to debug IPsec problems.

Procedure: To debug IPsec connections

Mode: Configure

Step	Command	Purpose
1	<code>node(cfg)#debug ipsec</code>	Enables IPsec debug monitor
2 optional	<code>node(cfg)#show ipsec security-associations</code>	Summarizes the configuration information of all IPsec connections. If an IPsec connection does not show up, then one or more parameters are missing in the respective Policy Profile. The information 'Bytes (processed)' supports debugging because it indicates whether IPsec packets depart from ('OUT') or arrive at ('IN') the SmartNode.

Example: IPsec Debug Output

```
SN(cfg)#debug ipsec
IPSEC monitor on
23:11:04 ipsec > Could not find security association for inbound ESP packet.
SPI:1201
```

Example: Display IPsec Security Associations

```
SN(cfg)#show ipsec security-associations

Active security associations:

Dir Type      Policy      Mode      Udp-Encapsulation
Peer          SPI AH      SPI ESP   AH          ESP-Auth   ESP-Enc
Bytes (processed/lifetime) Seconds (age/lifetime)
```

```

IN  MANUAL      ToBerne  Tunnel   no
200.200.200.1  -         1111    -         -         AES-CBC 128
3622/unlimited          19047/unlimited

OUT MANUAL      ToBerne  Tunnel   no
200.200.200.1  -         2222    -         -         AES-CBC 128
2857/unlimited          19047/unlimited

```

Sample configurations

The following sample configurations establish IPsec connections between a SmartNode and a Cisco router. To interconnect two SmartNodes instead, derive the configuration for the second SmartNode by doing the following modifications:

- Swap 'inbound' and 'outbound' settings
- Adjust the 'peer' setting
- Swap the private networks in the ACL profiles
- Adjust the IP addresses of the LAN and WAN interfaces
- Adjust the route for the remote network

IPsec tunnel, DES encryption

SmartNode configuration

```

profile ipsec-transform DES
  esp-encryption des-cbc 64

profile ipsec-policy-manual VPN_DES
  use profile ipsec-transform DES
  session-key inbound esp-encryption 1234567890ABCDEF
  session-key outbound esp-encryption FEDCBA0987654321
  spi inbound esp 1111
  spi outbound esp 2222
  peer 200.200.200.1
  mode tunnel

profile acl VPN_Out
  permit ip 192.168.1.0 0.0.0.255 172.16.0.0 0.0.255.255 ipsec-policy VPN_DES
  permit ip any any

profile acl VPN_In
  permit esp any any
  permit ah any any
  permit ip 172.16.0.0 0.0.255.255 192.168.1.0 0.0.0.255
  deny ip any any

context ip router

interface LAN
  ipaddress 192.168.1.1 255.255.255.0

interface WAN

```

```

    ipaddress 200.200.200.2 255.255.255.252
    use profile acl VPN_In in
    use profile acl VPN_Out out

context ip router
    route 0.0.0.0 0.0.0.0 200.200.200.1 0
    route 172.16.0.0 255.255.0.0 WAN 0

```

Cisco router configuration

```

crypto ipsec transform-set DES esp-des
!
crypto map VPN_DES local-address FastEthernet0/1
crypto map VPN_DES 10 ipsec-manual
    set peer 200.200.200.2
    set session-key inbound esp 2222 cipher FEDCBA0987654321
    set session-key outbound esp 1111 cipher 1234567890ABCDEF
    set transform-set DES
    match address 110
!
access-list 110 permit ip 172.16.0.0 0.0.255.255 192.168.1.0 0.0.0.255
!
interface FastEthernet0/0
    ip address 172.16.1.1 255.255.0.0
!
interface FastEthernet0/1
    ip address 200.200.200.1 255.255.255.252
    crypto map VPN_DES
!
ip route 192.168.1.0 255.255.255.0 FastEthernet0/1

```

IPsec tunnel, AES encryption at 256 bit key length, AH authentication with HMAC-SHA1-96

SmartNode configuration

```

profile ipsec-transform AES_SHA1
    esp-encryption aes-cbc 256
    ah-authentication hmac-sha1-96

profile ipsec-policy-manual VPN_AES_SHA1
    use profile ipsec-transform AES_SHA1
    session-key inbound ah-authentication 1234567890ABCDEF1234567890ABCDEF12345678
    session-key outbound ah-authentication FEDCBA0987654321FEDCBA0987654321FEDCBA09
    session-key inbound esp-encryption
1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF
    session-key outbound esp-encryption
FEDCBA0987654321FEDCBA0987654321FEDCBA0987654321FEDCBA0987654321
    spi inbound ah 3333
    spi outbound ah 4444
    spi inbound esp 5555
    spi outbound esp 6666
    peer 200.200.200.1
    mode tunnel
...

```

Rest of the configuration, see above, just change the name of the IPsec policy profile in the ACL profile `VPN_Out`

Cisco router configuration

```
crypto ipsec transform-set AES_SHA1 ah-sha-hmac esp-aes 256
!
crypto map VPN_AES_SHA1 local-address FastEthernet0/1
crypto map VPN_AES_SHA1 10 ipsec-manual
  set peer 200.200.200.2
  set session-key inbound esp 6666 cipher
FEDCBA0987654321FEDCBA0987654321FEDCBA0987654321FEDCBA0987654321
  set session-key outbound esp 5555 cipher
1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF
  set session-key inbound ah 4444 FEDCBA0987654321FEDCBA0987654321FEDCBA09
  set session-key outbound ah 3333 1234567890ABCDEF1234567890ABCDEF12345678
  set transform-set AES_SHA1
  match address 110
!
...
```

For the remainder of the configuration (see above), just change the name of the IPsec policy profile in the ACL profile `VPN_Out`

IPsec tunnel, 3DES encryption at 192 bit key length, ESP authentication with HMAC-MD5-96

SmartNode configuration

```
profile ipsec-transform TDES_MD5
  esp-encryption 3des-cbc 192
  esp-authentication hmac-md5-96

profile ipsec-policy-manual VPN_TDES_MD5
  use profile ipsec-transform TDES_MD5
  session-key inbound esp-authentication 1234567890ABCDEF1234567890ABCDEF
  session-key outbound esp-authentication FEDCBA0987654321FEDCBA0987654321
  session-key inbound esp-encryption
1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF
  session-key outbound esp-encryption
FEDCBA0987654321FEDCBA0987654321FEDCBA0987654321
  spi inbound esp 7777
  spi outbound esp 8888
  peer 200.200.200.1
  mode tunnel
...
```

For the remainder of the configuration (see above), just change the name of the IPsec policy profile in the ACL profile `VPN_Out`

Cisco router configuration

```
crypto ipsec transform-set 3DES_MD5 esp-3des esp-md5-hmac
!
crypto map VPN_3DES_MD5 local-address FastEthernet0/1
crypto map VPN_3DES_MD5 10 ipsec-manual
  set peer 200.200.200.2
```

```
    set session-key inbound esp 8888 cipher
    FEDCBA0987654321FEDCBA0987654321FEDCBA0987654321 authenticator
    FEDCBA0987654321FEDCBA0987654321
    set session-key outbound esp 7777 cipher
    1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF authenticator
    1234567890ABCDEF1234567890ABCDEF
    set transform-set 3DES_MD5
    match address 110
    !
    ...
```

For the remainder of the configuration (see above), just change the name of the IPsec policy profile in the ACL profile *VPN_Out*.

Chapter 26 **CS context overview**

Chapter contents

Introduction	294
CS context configuration task list	295
Planning the CS configuration	295
Configuring general CS settings.....	297
Configuring call routing	298
Creating and configuring CS interfaces.....	299
Specify call routing	299
Configuring dial tones	300
Configuring voice over IP parameters.....	300
Configuring ISDN ports	301
Configuring FXS ports	301
Configuring an H.323 VoIP connection	301
Configuring a SIP VoIP connection	301
Activating CS context configuration	302
Planning the CS context	305
Configuring general CS settings	306
Configuring call routing	306
Configuring VoIP settings	308
Configuring BRI ports	308
Configuring an H.323 VoIP connection	309
Activating the CS context configuration	309
Showing the running configuration	311

Introduction

This chapter gives an overview of SmartWare 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 SmartWare configuration concepts as described in chapter 2, “Configuration concepts” on page 35.

The CS context in SmartWare is a high level conceptual entity that is responsible for all aspects of circuit signaling, switching, and emulation. Besides the CS context itself, the CS entity consists of the following (indicated by the shaded area enclosed by a dashed line in figure 49):

- The CS interfaces
- ISDN and FXS ports
- Tone-set profiles
- SIP and H.323 gateways
- VoIP profiles

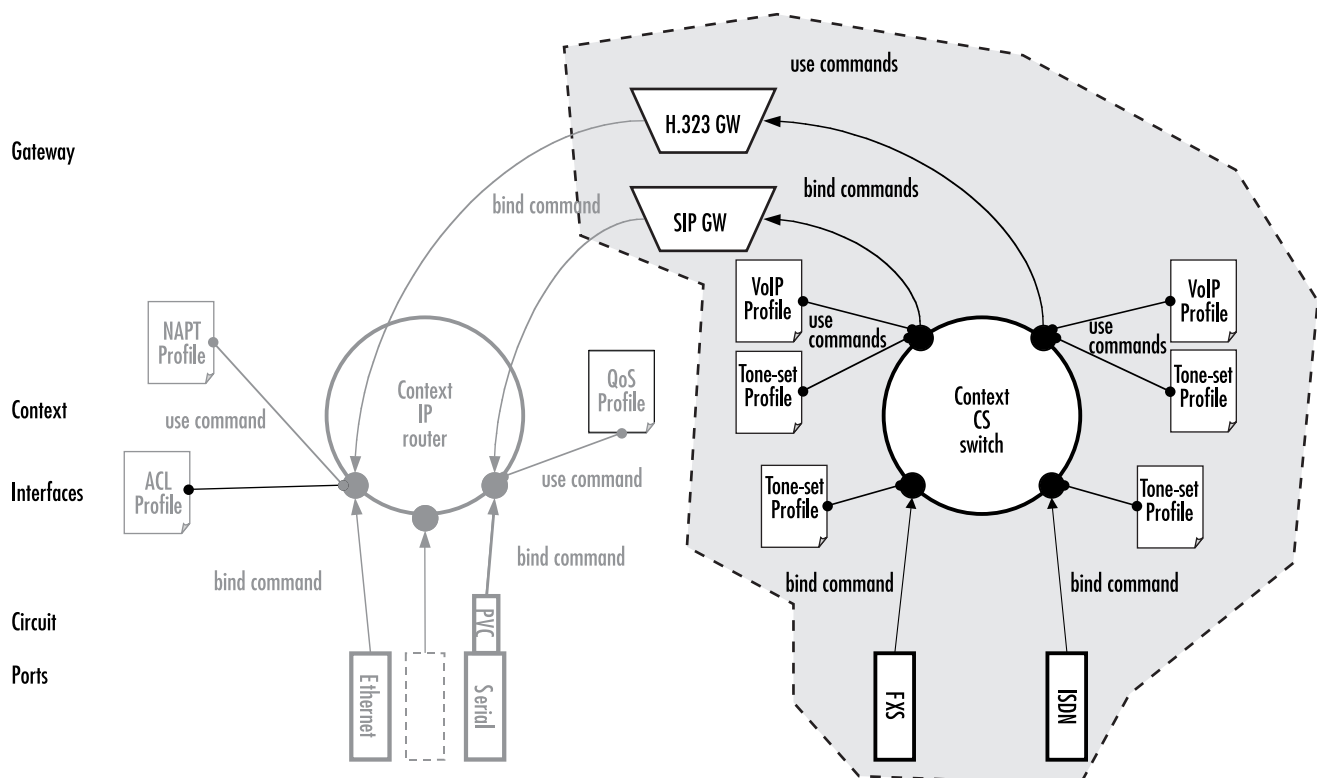


Figure 49. CS context configuration components

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 (ISDN, FXS)

interfaces and via VoIP interfaces (H.323, SIP) to the VoIP gateways and the IP context (see section “[Configuring call routing](#)” on page 298 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 298. These configuration tasks can be described in other chapters; thus, to configure call routing you have to refer to chapter 27, “[CS interface configuration](#)” on page 315 and chapter 33, “[Call router configuration](#)” on page 361.

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 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 task.

1. Planing the CS configuration
2. Configuring general CS settings
3. Configuring call routing
4. Configuring dial tones (advanced)
5. Configuring voice over IP settings (advanced)
6. Configuring ISDN ports
7. Configuring FXS ports
8. Configuring a H.323 VoIP connection
9. Configuring a SIP VoIP connection
10. Activating the CS context configuration

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 (H.323 gatekeeper settings or SIP registrar and proxy settings)
- Number and type of interface cards installed in your SmartNode
- Call routing

Figure 50 shows a typical application with a remote office in an enterprise network. The example focuses on the SmartNode 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 H.323 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.

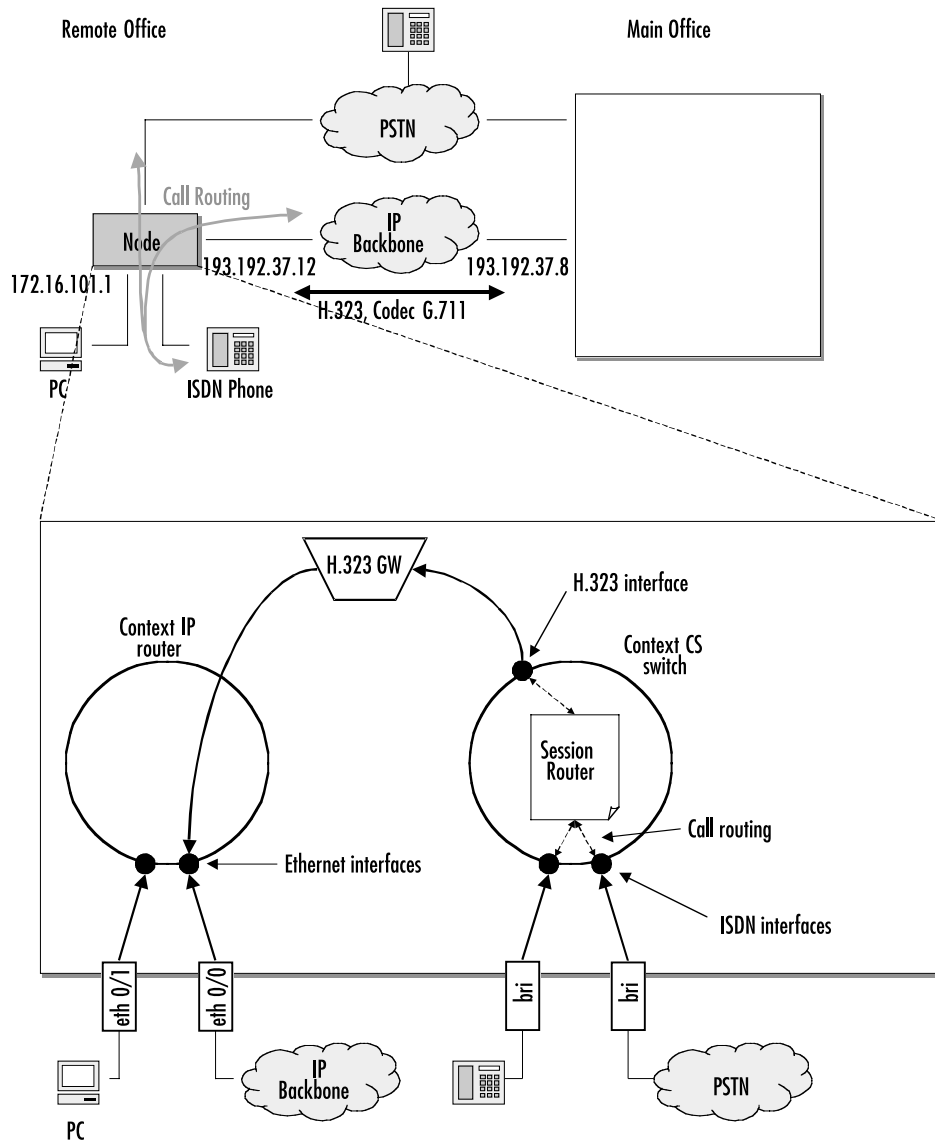


Figure 50. Remote office in an Enterprise network

An application like that shown in figure 50 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 297).

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 BRI ports will be needed, the first port for the ISDN phone and the second for the public ISDN network (see section “[Configuring ISDN ports](#)” on page 301).
- Two ISDN interfaces will be needed, each bound to a BRI port (see section “[Configuring call routing](#)” on page 298)
- An H.323 interface is required in order to use H.323 (see section “[Configuring call routing](#)” on page 298)
- The call router routing tables, and the H.323 and ISDN interfaces will have to be configure to support call routing (see section “[Configuring call routing](#)” on page 298).

Calls are routed from an ISDN phone 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 phone to the PSTN and from the PSTN or main office to the ISDN phone.

- The H.323 gateway must be configured to use the G.711 codec (see section “[Configuring an H.323 VoIP connection](#)” on page 301)
- 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 SmartWare CS configuration. The IP configuration is not a topic in this example. For more information on IP configuration refer to chapter 10, “[IP context overview](#)” on page 117.

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 SmartNode. In most cases, the default value is acceptable, so that you do not need to perform these configuration tasks:

- **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). If you do have a connection to a public ISDN, select that as your source for the reference clock.
- **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

Mode: System

Step	Command	Purpose
1	<code>node(sys)#clock-source internal</code> or <code>node (sys)#clock-source slot-number port-number</code>	Generates the reference clock internally or specifies a specific port to receive the reference clock.
2	<code>node (sys)#ic voice slot-number</code>	Changes to ic_voice mode.
3	<code>node (ic-voice)[slot-number]#pcm law-select { aLaw uLaw }</code>	Selects the PCM aLaw for Europe or uLaw for USA.

Configure: General CS settings

The following example configures the general CS parameters

```
SN>enable
SN#configure
SN(cfg)#system
SN(sys)#clock-source 1 0
SN(sys)#ic voice 1
SN(ic-voice)[1]#pcm law-select aLaw
SN(ic-voice)[1]#exit
```

Configuring call routing

Calls through a SmartNode 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 33, “[Call router configuration](#)” on page 361.

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. There are four types of CS interfaces: two PSTN (ISDN and FXS) and two VoIP (H.323 and SIP). The differences between these interfaces are explained later.

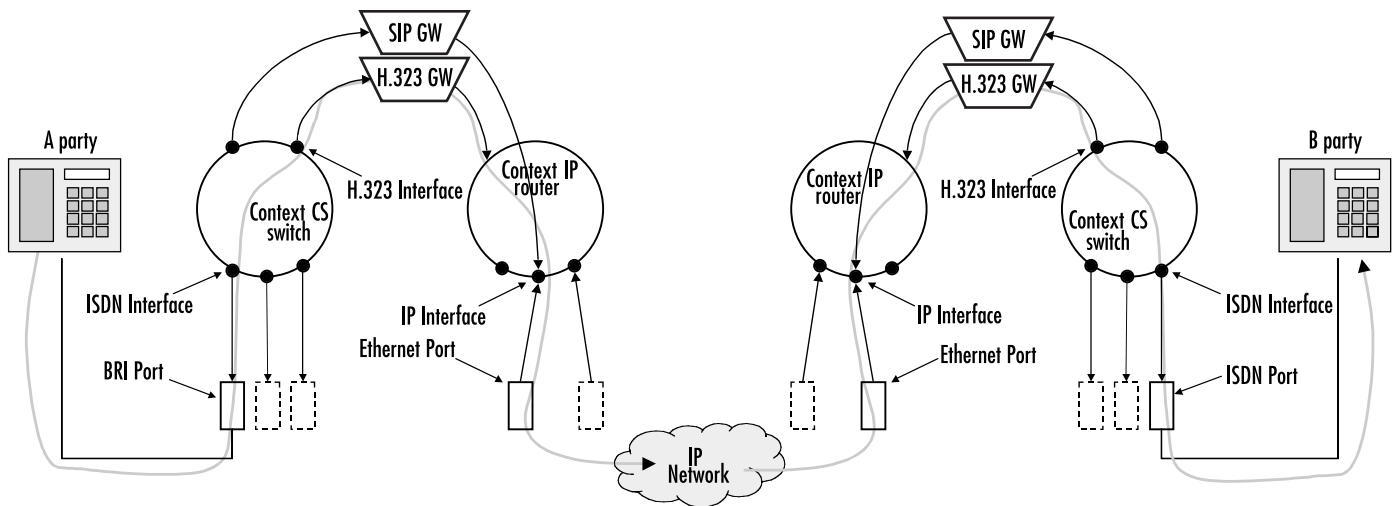


Figure 51. Direct call routing from one SmartNode to another

Figure 51 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 H.323 interface. Once it has passed the IP context and the IP network, the other SmartNode—from the H.323 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 creating call router tables.

Creating and configuring CS interfaces

SmartWare supports multiple instances of the CS context. 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 (H.323 and SIP) are bound to a gateway. ISDN and FXS ports are bound up to a PSTN interface (ISDN and FXS, respectively).

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 27, “CS interface configuration” on page 315.

Specify call routing

As mentioned previously, for basic call routing you can omit creating call router tables. SmartWare 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 27, “[CS interface configuration](#)” on page 315. 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 33, “[Call router configuration](#)” on page 361. In this chapter, the differences between basic interface routing and advanced call routing are described in more detail.

Configuring dial tones

SmartWare 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 SmartWare, there are many configurable parameters that can affect a voice over IP connection. SmartWare supports the H.323 and the SIP protocols that transmit voice packets over IP.

The *voice over IP* (VoIP) parameters are configured in the VoIP profile. A VoIP profile is used by a H.323 or SIP interface. All calls going through that interface (see [figure 51](#) on page 299) 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 40, “[VoIP profile configuration](#)” on page 475 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 that you generally do not need to perform these configuration tasks.

If no VoIP profile is specified to be used 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 SmartNode. 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 35, “[ISDN port configuration](#)” on page 425.

Configuring FXS ports

FXS ports represents physical ports on the SmartNode. To configure the FXS ports, refer to chapter 36, “[FXS port configuration](#)” on page 441.

Configuring an H.323 VoIP connection

To configure a H.323 connection, you have to specify the voice codec selection used for the VoIP profile and the call signaling.

Configuring the voice codec for an H.323 connection is done on a H.323 interface by specifying the VoIP profile that shall be used. The VoIP profile contains an ordered list of codecs that must be used for codec negotiation for all calls that pass this interface. During a call setup, the first codec that is specified in the VoIP profile is taken. For information how to configure the codecs, refer to chapter 40, “[VoIP profile configuration](#)” on page 475.

H.323 offers *direct call signaling* and *gatekeeper routed call signaling* methods. For direct call signaling, you have to specify the remote terminal or gateway on each H.323 interface. Gatekeeper routed call signaling uses a gatekeeper to find the destination address. For examples and information on how to configure direct call signaling on H.323 voice connections, refer to chapter 31, “[H.323 interface configuration](#)” on page 347. To configure gatekeeper routed call signaling on H.323 voice connections, refer to chapter 38, “[H.323 gateway configuration](#)” on page 451.

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.

Configuring the voice codec for a SIP connection is similar to the H.323 connection. You have to 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 that pass 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 40, “[VoIP profile configuration](#)” on page 475.

You can configure the SIP gateway to register to a registrar with multiple URIs. Optionally, you can configure the 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 SIP gateway, refer to chapter 39, “[SIP gateway configuration](#)” on page 465. To configure SIP interfaces, refer to chapter 32, “[SIP interface configuration](#)” on page 355.

Activating CS context configuration

After configuring the CS context and its components, the configuration must be activated. This includes binding the physical ports (BRI, E1, T1 or FXS) to the CS interfaces (ISDN and FXS) and enabling the gateways, ports, and the CS context.

In order to become functional, each ISDN or FXS interface must be bound from one BRI, E1, T1, or FXS port from which it receives incoming calls, and to which it forwards outgoing calls. Unlike ISDN and FXS interfaces, VoIP interfaces (H.323 and SIP interfaces) must be bound to a gateway.

Note The difference between VoIP and PSTN interface is that VoIP interfaces are bound to a gateway while PSTN ports are bound to a CS interface. After binding to become active, the BRI, E1, T1 or FXS port must be enabled.

To bind an ISDN port to an ISDN interface, refer to chapter 35, “[ISDN port configuration](#)” on page 425. To bind an FXS port to an FXS interface, refer to chapter 36, “[FXS port configuration](#)” on page 441. Likewise, the H.323 or SIP gateway must be enabled. Additionally, the H.323 or SIP gateway must be bound to a specific IP interface. For more information, refer to chapter 38, “[H.323 gateway configuration](#)” on page 451 or chapter 39, “[SIP gateway configuration](#)” on page 465.

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. SmartWare 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 42, “[VoIP debugging](#)” on page 497 for an introduction to the configuration debugging possibilities in SmartWare.

Note You can modify the configuration at runtime; changes will be active after 3 seconds. It is not necessary to shutdown 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. Currently open 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 27, “[CS interface configuration](#)” on page 315” and chapter 33, “[Call router configuration](#)” on page 361.

Procedure: Show the CS context configuration, enable the call router debug monitor and activate the CS context

Mode: Context CS

Step	Command	Purpose
1	node(ctx-cs)[switch]#show call-router config detail level	Show the CS context configuration. <i>Level</i> could be 1..5. Level 1 shows less, level 5 shows all information.
2	node (ctx-cs)[switch]#debug call-router detail level	Enable 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	node (ctx-cs)[switch]#no shutdown	Enable the CS context, checks the interface and call router configuration
4	node(ctx-cs)[switch]#show call-router status detail level	Show 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.

```
SN(cfg)#show call-router config detail 5
Table switch/TAB-ISDN-SERVICE:
Key          Value          Function          Dest-Type          Dest-Name
itc          -
-----
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  -
-----
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        -                -                -

...

SN(cfg)#debug call-router
SN(cfg)#context cs
SN(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-NODE-B-precallservice'
02:14:33 CR > [switch] Loading table 'IF-NODE-C-precallservice'
SN(ctx-cs)[switch]#

```

Example: Configure SmartNode in an Enterprise Network

Situation: Figure 52 shows an enterprise network with a SmartNode 2300 configured with a BRI interface card in slot 2. A PBX, a LAN, the PSTN, and the company network are connected. The VoIP protocol used is H.323. There is no gatekeeper, so direct call signaling is used. 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.

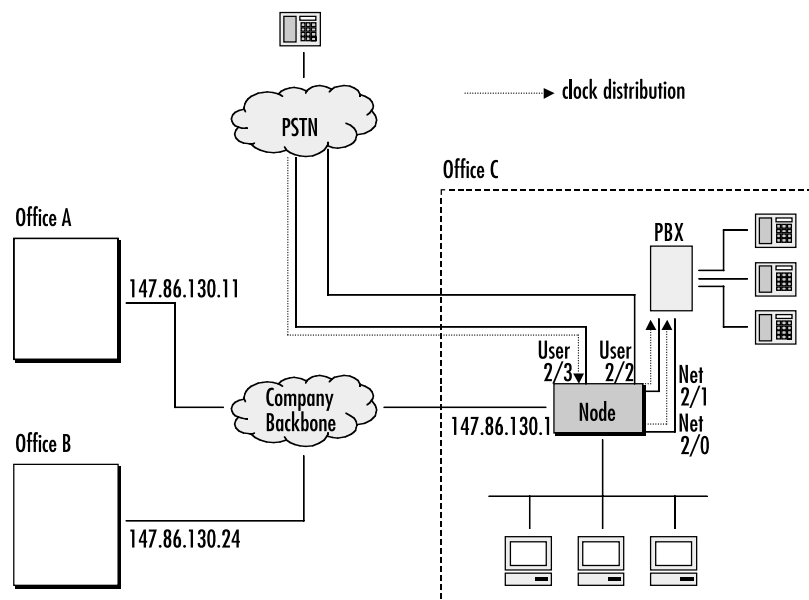


Figure 52. SmartNode in an Enterprise network

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)

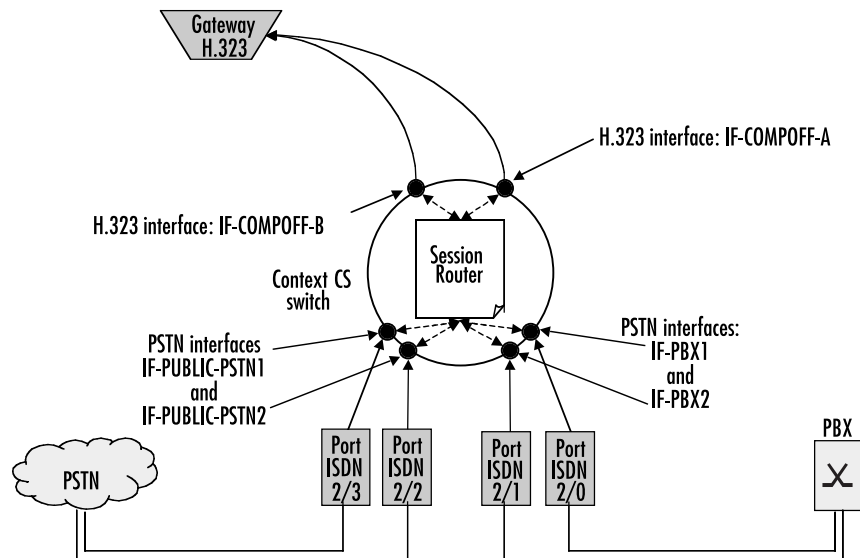


Figure 53. CS Configuration

Planning the CS context

Based on the criteria used in the previous example, the following configuration information applies (see figure 53):

- 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 301).
- 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.
- For every remote H.323 device we need a H.323 interface. There are two in total. One gets the remote IP address of the SmartNode in office A, the other the IP address of the SmartNode in office B. (Refer to section “Configuring call routing” on page 298).
- 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 298).
- 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 300).
- We define H.323 'direct call signaling'. (Refer to section “Configuring an H.323 VoIP connection” on page 301).

Configuring general CS settings

First we set clock-source to ISDN port 2/3.

```
SN>enable
SN#configure
SN(cfg)#system
SN(sys)#clock-source 2 3
SN(sys)#exit
SN(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:

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface isdn IF-PBX1
SN(if-pstn)[IF-PBX1]#route call dest-table TAB-CALLED-NUMBER
SN(if-pstn)[IF-PBX1]#exit

SN(ctx-cs)[switch]#interface isdn IF-PBX2
SN(if-pstn)[IF-PBX2]#route call dest-table TAB-CALLED-NUMBER
SN(if-pstn)[IF-PBX2]#exit

SN(ctx-cs)[switch]#interface isdn IF-PUBLIC-PSTN1
SN(if-pstn)[IF-PUBL~]#route call dest-table TAB-CALLED-NUMBER
SN(if-pstn)[IF-PUBL~]#exit

SN(ctx-cs)[switch]#interface isdn IF-PUBLIC-PSTN2
SN(if-pstn)[IF-PUBL~]#route call dest-table TAB-CALLED-NUMBER
SN(if-pstn)[IF-PUBL~]#exit
SN(ctx-cs)[switch]#
```

In addition, we create the two H.323 interfaces and configure call routing, as well as the IP address of the remote H.323 terminal, which is the IP address of the device in office A or office B, respectively.

```
SN(ctx-cs)[switch]#interface h323 IF-COMPOFF-A
SN(if-h323)[IF-COMP~]#route call dest-table TAB-CALLED-NUMBER
SN(if-h323)[IF-COMP~]#remoteip 146.86.130.11
SN(if-h323)[IF-COMP~]#bind gateway h323
SN(if-h323)[IF-COMP~]#exit

SN(ctx-cs)[switch]#interface h323 IF-COMPOFF-B
SN(if-h323)[IF-COMP~]#route dest-table calledNumberRouting
SN(if-h323)[IF-COMP~]#remoteip 146.86.130.24
SN(if-h323)[IF-COMP~]#bind gateway h323
SN(if-h323)[IF-COMP~]#exit
SN(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:

```
SN(ctx-cs)[switch]#routing-table called-e164 TAB-CALLED-NUMBER
SN(rt-tab)[TAB-CAL~]#route 1.. dest-service HUNT-COMPOFF-A
SN(rt-tab)[TAB-CAL~]#route 2.. dest-service HUNT-COMPOFF-B
SN(rt-tab)[TAB-CAL~]#route 5.. dest-service HUNT-PBX
SN(rt-tab)[TAB-CAL~]#route default dest-service HUNT-PUBLIC-PSTN
SN(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
SN(rt-tab)[TAB-CAL~]#exit
SN(ctx-cs)[switch]#
```

The hunt group HUNT-COMPOFF-A tries to reach the company office A routing the call directly to the H.323 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 H.323 interface IF-COMPOFF-B first.

```
SN(ctx-cs)[switch]#service hunt-group HUNT-COMPOFF-A
SN(rt-tab)[HUNT-CO~]#no cyclic
SN(rt-tab)[HUNT-CO~]#timeout 5
SN(rt-tab)[HUNT-CO~]#route call 1 dest-interface IF-COMPOFF-A
SN(rt-tab)[HUNT-CO~]#route call 2 dest-service HUNT-PUBLIC-PSTN
SN(rt-tab)[HUNT-CO~]#exit
SN(ctx-cs)[switch]#service hunt-group HUNT-COMPOFF-B
SN(rt-tab)[HUNT-CO~]#no cyclic
SN(rt-tab)[HUNT-CO~]#timeout 5
SN(rt-tab)[HUNT-CO~]#route call 1 dest-interface IF-COMPOFF-B
SN(rt-tab)[HUNT-CO~]#route call 2 dest-service HUNT-PUBLIC-PSTN
SN(rt-tab)[HUNT-CO~]#exit
SN(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.

```
SN(ctx-cs)[switch]#service hunt-group HUNT-PBX
SN(rt-tab)[HUNT-PB~]#cyclic
SN(rt-tab)[HUNT-PB~]#route call 1 dest-interface IF-PBX1
SN(rt-tab)[HUNT-PB~]#route call 2 dest-interface IF-PBX2
SN(rt-tab)[HUNT-PB~]#exit
SN(ctx-cs)[switch]#service hunt-group HUNT-PUBLIC-PSTN
```

```

SN(rt-tab)[HUNT-PU~]#cyclic
SN(rt-tab)[HUNT-PU~]#route call 1 dest-interface IF-PUBLIC-PSTN1
SN(rt-tab)[HUNT-PU~]#route call 2 dest-interface IF-PUBLIC-PSTN2
SN(rt-tab)[HUNT-PU~]#exit
SN(ctx-cs)[switch]#exit
SN(cfg)#

```

Configuring VoIP settings

Because we need G.723 as codec we enable DTMF relay:

```

SN(cfg)#profile voip H323-VOIP-PROFILE
SN(pf-voip)[H323-VO~]#codec 1 g723-6k3
SN(pf-voip)[H323-VO~]#dtmf-relay
SN(pf-voip)[H323-VO~]#exit
SN(cfg)#

```

We want to use this profile on our H.323 interfaces:

```

SN(cfg)#context cs
SN(ctx-cs)[switch]#interface h323 IF-COMPOFF-A
SN(if-h323)[IF-COMP~]#use profile voip H323-VOIP-PROFILE
SN(if-h323)[IF-COMP~]#exit
SN(ctx-cs)[switch]#interface h323 IF-COMPOFF-B
SN(if-h323)[IF-COMP~]#use profile voip H323-VOIP-PROFILE
SN(if-h323)[IF-COMP~]#exit
SN(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:

```

SN(cfg)#port bri 2 0
SN(prt-bri)[2/0]#q921
SN(q921)[2/0]#protocol pp
SN(q921)[2/0]#q931
SN(q931)[2/0]#uni-side net
SN(q931)[2/0]#encapsulation cc-isdn
SN(q931)[2/0]#bind interface IF-PBX1
SN(q931)[2/0]#exit
SN(q921)[2/0]#exit
SN(prt-bri)[2/0]#no shutdown
SN(cfg)#port bri 2 1
SN(prt-bri)[2/1]#q921
SN(q921)[2/1]#protocol pp
SN(q921)[2/1]#q931
SN(q931)[2/1]#uni-side net
SN(q931)[2/1]#encapsulation cc-isdn
SN(q931)[2/1]#bind interface IF-PBX1
SN(q931)[2/1]#exit
SN(q921)[2/1]#exit
SN(prt-bri)[2/1]#no shutdown
SN(cfg)#port bri 2 2
SN(prt-bri)[2/2]#q921
SN(q921)[2/2]#protocol pp
SN(q921)[2/2]#q931

```

```

SN(q931)[2/2]#uni-side user
SN(q931)[2/2]#encapsulation cc-isdn
SN(q931)[2/2]#bind interface IF-PBX1
SN(q931)[2/2]#exit
SN(q921)[2/2]#exit
SN(prt-bri)[2/2]#no shutdown
SN(cfg)#port bri 2 1
SN(prt-bri)[2/3]#q921
SN(q921)[2/3]#q931
SN(q921)[2/3]#protocol pp
SN(q931)[2/3]#uni-side user
SN(q931)[2/3]#encapsulation cc-isdn
SN(q931)[2/3]#bind interface IF-PBX1
SN(q931)[2/3]#exit
SN(q921)[2/3]#exit
SN(prt-bri)[2/3]#no shutdown

```

Configuring an H.323 VoIP connection

Next we configure call signaling:

```

SN(cfg)#gateway h323 h323
SN(gw-h323)[h323]#no ras
SN(gw-h323)[h323]#faststart
SN(gw-h323)[h323]#bind interface eth0
SN(gw-h323)[h323]#exit
SN(cfg)#

```

Activating the CS context configuration

Prior to activating our configuration we use two 'show' commands to display part of our configuration:

```

SN(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  -              -                dest-service       HUNT-COMPOFF-A
1..         -              -                dest-service       HUNT-COMPOFF-B
2..         -              -                dest-service       HUNT-PBX
5..         -              -                dest-service       HUNT-PUBLIC-PSTN
default     -              -                dest-service       HUNT-PUBLIC-PSTN
SN(cfg)#
SN(cfg)#show gateway h323 config detail 5
H.323 Gateway: h323
=====

RAS Engine
-----

Administrative Status:          no
Gatekeeper-Discovery:         auto
Gatekeepers
Re-Registration Time:          90s
Local Aliases
Source Information

Faststart:                      yes
Early-H.245:                    no
H.245-Tunneling:                no
Call-Signaling:                 147.86.130.1/1720
Administrative Status:         close
SN(cfg)#

```

Finally, activate the gateway and CS context:

```

SN(cfg)#gateway h323
SN(gw-h323)[gw_name]#no shutdown
SN(gw-h323)[gw_name]#exit
SN(cfg)#debug call-router detail 5
SN(cfg)#context cs
SN(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'
SN(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 H323-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 h323 IF-COMPOFF-A
  bind gateway h323
  route call dest-table TAB-CALLED-NUMBER
  remoteip 146.86.130.11
  use profile voip H323-VOIP-PROFILE

interface h323 IF-COMPOFF-A
  bind gateway h323
  route call dest-table TAB-CALLED-NUMBER
  remoteip 146.86.130.24
  use profile voip H323-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

gateway h323 h323
faststart
bind interface eth0 router
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
encapsulation q931

q931
protocol dss1
uni-side net
encapsulation cc-isdn
bind interface IF-PBX1

port bri 2 0
no shutdown

port bri 2 1
clock auto
encapsulation q921

q921
protocol pp
uni-side auto
encapsulation q931

q931
protocol dss1
uni-side net
encapsulation cc-isdn
bind interface IF-PBX2
```

```
port bri 2 1
  no shutdown

port bri 2 2
  clock auto
  encapsulation q921

q921
  protocol pp
  uni-side auto
  encapsulation q931

q931
  protocol dss1
  uni-side user
  encapsulation cc-isdn
  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
  encapsulation q931

q931
  protocol dss1
  uni-side user
  encapsulation cc-isdn
  bind interface IF-PUBLIC-PSTN2

port bri 2 3
  no shutdown
```

Chapter 27 **CS interface configuration**

Chapter contents

Introduction	316
CS interface configuration task list	317
Creating and configuring CS interfaces.....	317
Configuring call routing	318
Configuring the interface mapping tables	319
Configuring the precall service tables	322

Introduction

This chapter provides an overview of interfaces in the CS context and describes the tasks involved in their configuration. Within the CS context of the SmartWare, an interface is a logical entity providing call signaling for incoming and outgoing calls to and from ISDN or FXS 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 (SmartWare) 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 54 illustrates the function of the CS interfaces. The types of CS interfaces are:

- PSTN interfaces (ISDN and FXS). An ISDN interface is used to access the ISDN stack that runs on top of an ISDN port. Layer 3 (Q.931) of the ISDN stack is bound to this CS interface. Incoming calls from the ISDN stack are routed by the call router according the route configuration on the ISDN interface. Likewise, an FXS interface handles an FXS port, which must be bound to the FXS interface. Binding is done from a port to an interface.
- VoIP interface (H.323 and SIP). H.323 and SIP interfaces are CS interface types that provide voice over IP settings in addition to the general CS interface parameters. These interfaces must be explicitly bound to an existing H.323 or SIP gateway, respectively.

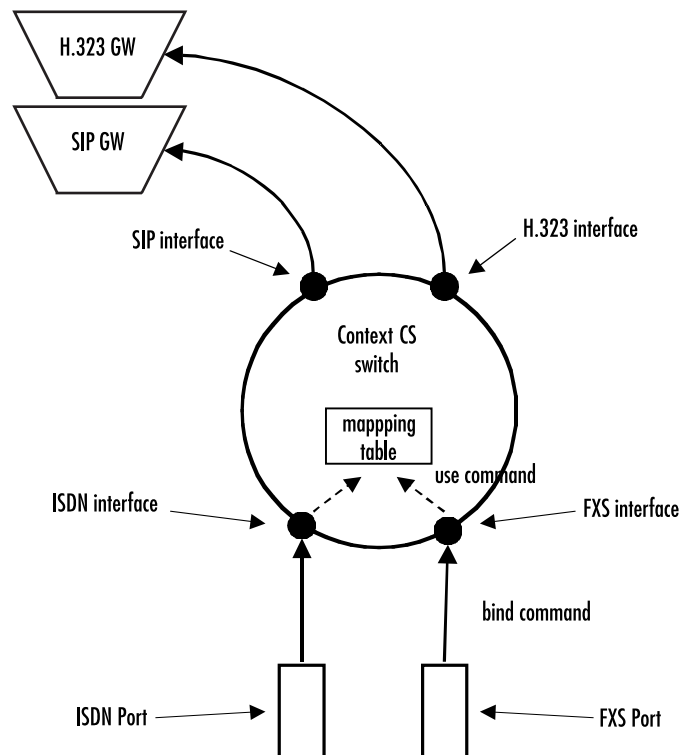


Figure 54. CS interfaces on the CS context

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 34, “Tone configuration” on page 417 and chapter 40, “VoIP profile configuration” on page 475. 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 <i>if-type</i> <i>if-name</i>	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 h323 , sip , isdn and fxs .
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		Repeat step 1 to 5 to create and configure your CS interfaces
7	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.
8	node(ctx-cs)[switch]#no interface <i>if-type</i> <i>if-name</i>	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.

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface isdn IF-PBX1
SN(if-pstn)[IF-PBX1]#route call dest-interface TAB-CALLED-NUMBER
SN(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
SN(if-pstn)[IF-PBX1]#exit
SN(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
SN(ctx-cs)[switch]#no interface isdn IF-PBX1
SN(ctx-cs)[switch]#
```

Configuring call routing

SmartWare 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 33, “Call router configuration” on page 361, 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
4		Repeat steps 1–3 for all the required CS interfaces

Example: Configure call routing

The following example shows how to configure basic interface routing.

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface pstn IF-PBX1
SN(if-pstn)[IF-PBX1]#route call dest-interface IF-H323-0
SN(if-pstn)[IF-PBX1]#exit
SN(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 at the same time, called and calling party properties!

You can choose 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 33, “[Call router configuration](#)” on page 361 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</i> <i>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> <i>and/or</i> 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.

```
SN(ctx-cs)[switch]#mapping-table e164 to e164 PRIV-TO-GLOB
SN(map-tab)[PRIV-TO~]#map (..) to 00419988825\1
SN(map-tab)[PRIV-TO~]#exit
SN(ctx-cs)[switch]#mapping-table e164 to e164 GLOB-TO-PRIV
SN(map-tab)[GLOB-TO~]#map 00419988825(..) to \1
SN(map-tab)[GLOB-TO~]#exit
SN(ctx-cs)[switch]#interface isdn IF-PHONES
SN(if-isdn)[IF-PHON~]#route dest-table TAB-CALLED-NUMBER
SN(if-isdn)[IF-PHON~]#use mapping-table in PRIV-TO-GLOB
SN(if-isdn)[IF-PHON~]#use mapping-table out GLOB-TO-PRIV
SN(if-isdn)[IF-PHON~]#exit
SN(ctx-cs)[switch]#
```

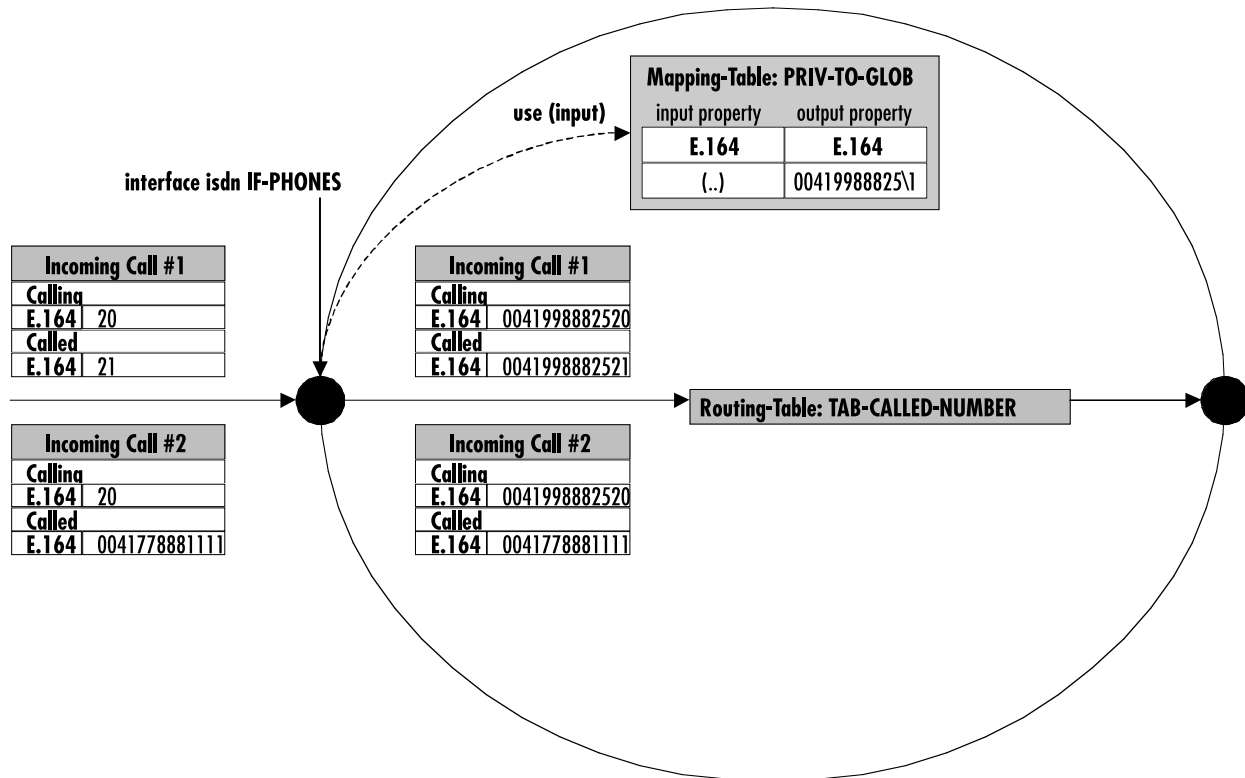


Figure 55. Incoming call passing an interface mapping table

Figure 55 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. Which 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.

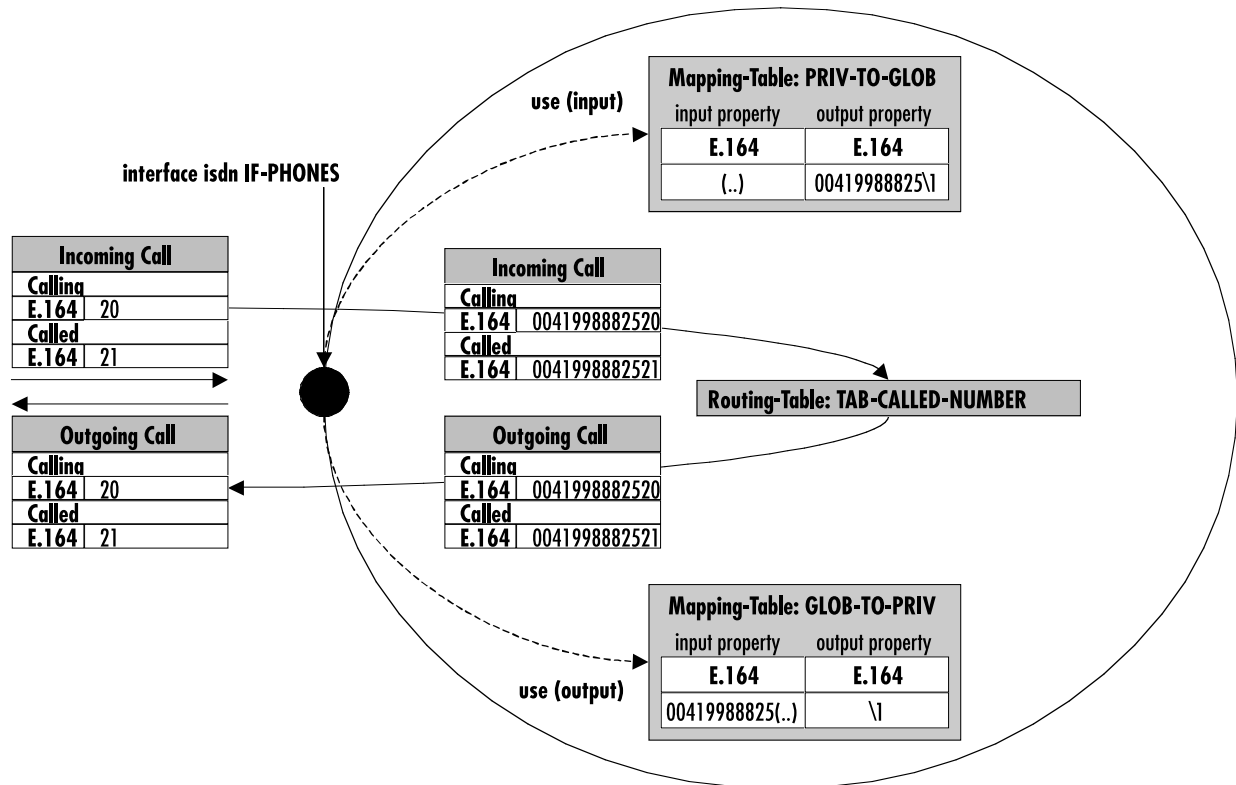


Figure 56. Call passing an input and an output mapping table

Let's assume we manipulate an incoming ISDN call using the PRIV-TO-GLOB mapping table as in the previous example. Figure 56 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 mapping tables are used to convert dialed special numbers like *61 to invocation commands for supplementary services like call-waiting, etc. Precall service tables are configured as part of the call router in the context CS configuration mode. Precall service tables are used on an FXS interface where the attached phone should be able to activate or deactivate services by dialing a special number. SmartWare currently supports the following service commands:

- **activate-cw**—Activates call-waiting on the interface that uses the precall service table. Once activated a second incoming call is possible on the interface. The second call is announced to the first call. The user can then 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.

Note Currently you can only use precall service tables on FXS interfaces.

Procedure: To create precall service table and use it on an FXS interface

Mode: Context CS

Step	Command	Purpose
1	node(ctx-cs)[switch]#precall-service-table <i>table-name</i>	Creates a new table that maps special numbers into supplementary service invocation commands
2	node(pcs-tab)[table-name]#map <i>special-number to command</i>	Adds a new entry to map a <i>special-number</i> into a supplementary service invocation <i>command</i> .
3		Repeat Step 2 to add other special number mappings.
4	node(pcs-tab)[table-name]#exit	Returns to context CS Configuration Mode
5	node(ctx-cs)[switch]#interface fxs <i>if-name</i>	Enters FXS Interface Configuration Mode of interface <i>if-name</i>
6	node(if-fxs)[if-name]#use mapping-table precall-service <i>table-name</i>	Uses the precall service table created with step 1 to 4 on this FXS interface.

Example: Create and use a precall service table

The following example shows how to create a precall service table that treats *43# as activation command for the call-waiting supplementary service, while #43# is used to deactivate call-waiting and *#43# is used to query the call-waiting supplementary service:

```
SN(ctx-cs)[switch]#precall-service-table SUPP-SVC
SN(pcs-tab)[SUPP-SVC]#map *43# to activate-cw
SN(pcs-tab)[SUPP-SVC]#map #43# to deactivate-cw
SN(pcs-tab)[SUPP-SVC]#map *#43# to interrogate-cw
SN(pcs-tab)[SUPP-SVC]#exit
SN(ctx-cs)[switch]#interface fxs IF-PHONE
SN(if-fxs)[IF-PHONE]#use mapping-table precall-service SUPP-SVC
SN(if-fxs)[IF-PHONE]#exit
SN(ctx-cs)[switch]#
```


Chapter 28 **ISDN interface configuration**

Chapter contents

Introduction	326
ISDN interface configuration task list.....	326
Configuring DTMF dialing (optional)	326
Configuring an alternate PSTN profile (optional)	327

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 27, “[CS interface configuration](#)” on page 315)

An ISDN interface represents the connection of an ISDN signaling channel to the call control of SmartWare. 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 57](#)).

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. See chapter 35, “[ISDN port configuration](#)” on page 425 for details.

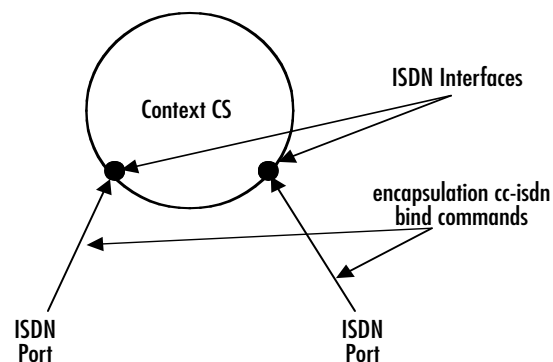


Figure 57. ISDN interfaces on the CS context

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 DTMF dialing (optional)
- Using an alternate PSTN profile (optional)

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 the use with such 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	node(if-ISDN)[if-name]#[no] dtmf-dialing	Enables/Disables DTMF dialing (default: disabled)

Example: Enable DTMF dialing

The following example shows how to enable DTMF dialing for a given ISDN interface.

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface isdn MyIsdnIf
SN(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 41, “PSTN profile configuration” on page 493). 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	node(if-isdn)[if-name]#[no] use profile pstn profile-name	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*.

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface isdn myIsdnIf
SN(if-isdn)[myIsdnIf]#use profile pstn myprofile
```


Chapter 29 **FXS interface configuration**

Chapter contents

Introduction	330
FXS supplementary services description	330
Call holding	330
Call waiting	330
Additional call offering	331
FXS interface configuration task list	331
Configuring a subscriber number (recommended)	331
Configuring an alternate PSTN profile (optional)	332
Configuring caller-ID presentation (optional)	332
Configuring call holding supplementary service (optional)	333
Configuring call waiting supplementary service (optional)	333
Configuring additional call offering supplementary service (optional)	334

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 equal to all CS interfaces. Information about basic interface configuration can be found in the general chapter about CS interface configuration (see chapter 27, “CS interface configuration” on page 315).

An FXS interface represents the connection of an analog FXS port signaling to the call control of SmartWare. It encapsulates the signaling of the exchange side of a POTS line, allows incoming and outgoing calls on this line, controls the line events, tones and datapath, and provides a set of supplementary 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 58).

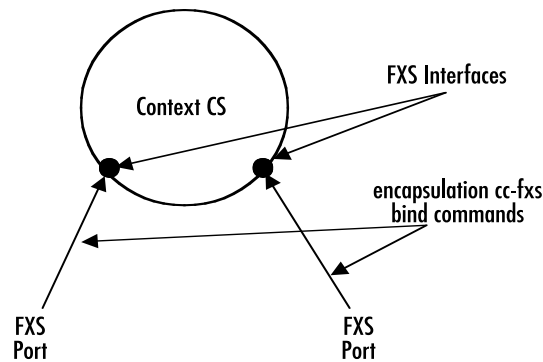


Figure 58. FXS interfaces on the CS context

FXS supplementary services description

FXS interfaces offer a set of supplementary services. These services are locally terminated (i.e. no other device is involved in providing the services), and they can be enabled/disabled separately. The services are:

- Call holding
- Call waiting
- Additional call offering

Other supplementary services can be configured using general context CS functionality.

Call holding

There is the possibility to have two open calls on one POTS terminal: One of the two calls being active, the other passive or *on hold*.

Procedure: To toggle active and passive call, press flash-hook, followed by the “2” key

Call waiting

When the analog line is busy, a second incoming call on the same interface announces itself using a special tone, the *waiting tone*. The user can then decide whether to accept the new call (put the current on hold or drop it), or to reject it (keep the current call).

This feature is not used when the connected analog equipment is a fax, answering machine or similar device.

Procedure: To reject a waiting call, when a waiting call is announced with a special tone signal, the user can either:

- Press flash-hook, followed by the “0” key
- Ignore the call waiting signal

The waiting call will be rejected.

Procedure: To accept a waiting call, when a waiting call is announced with a special tone signal, the user can:

- Press flash-hook, followed by the “2” key to put the current call on hold and accept the waiting call. Call hold service must be enabled for this to work.
- Press flash-hook, followed by the “1” key to terminate the current call and accept the waiting call
- Go on-hook to terminate the current call. The terminal will ring to indicate that a call is waiting—when this call is accepted by going off-hook again, the waiting call is connected.

Additional call offering

To issue a callback during a call, an additional call can be offered, putting the current call on hold.

Procedure: To make a second call when yet in a call, press flash-hook, and wait for the dialtone. Then dial the number. This places the current call on hold and makes the new call active.

To toggle between the active and the held call, press flash-hook, followed by the “2” key. Call holding service must be enabled for this to work.

FXS interface configuration task list

This section describes the configuration tasks for FXS interfaces. There is no mandatory configuration for basic FXS operation—most configuration commands refer to the use of supplementary services.

Next to the basic CS interface settings, the following configurations can be made:

- Configuring a subscriber number (recommended)
- Using an alternate PSTN profile (optional)
- Configuring caller-id presentation (optional)
- Configuring call holding supplementary service (optional)
- Configuring call waiting supplementary service (optional)
- Configuring additional call offering supplementary service (optional)

Configuring a subscriber number (recommended)

Contrary to ISDN, where each terminal knows its own subscriber number (MSN), an analog device doesn't have this capability. If such a device 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 POTS protocol. To insert calling party information and make it available to other protocols over which the call may be transported, a subscriber number must be configured on the interface.

Note The configured subscriber number does not affect the routing of incoming calls on the interface.

Procedure: To configure a subscriber number for calls originating from the interface

Mode: Interface FXS

Step	Command	Purpose
1	<code>node(if-fxs)[if-name]#[no] subscriber-number</code>	Sets/Removes a subscriber number

Example: Set the subscriber number

The following example shows how to set the subscriber number.

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface fxs myFxsIf
SN(if-fxs)[myFxsIf]#subscriber-number 110
```

Configuring an alternate PSTN profile (optional)

The PSTN profile contains the configuration for data/voice transmission on circuit-switched channels (see chapter 41, “PSTN profile configuration” on page 493). In the case of FXS interfaces, the PSTN profile applies to the analog line 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 FXS interface, the profile named *default* is used.

Procedure: To define an alternate PSTN profile for the FXS interface

Mode: Interface FXS

Step	Command	Purpose
1	<code>node(if-fxs)[if-name]#[no] use profile pstn profile-name</code>	Defines an alternate PSTN profile to be used for this FXS 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 named *default* of the FXS interface by an alternate PSTN profile named *myprofile*.

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface fxs myFxsIf
SN(if-fxs)[myFxsIf]#use profile pstn myprofile
```

Configuring caller-ID presentation (optional)

POTS protocols allow the presentation of the caller-ID (calling party number of an incoming call) to an analog terminal when the terminal is ringing. (See also chapter 36, “FXS port configuration” on page 441 for other caller-ID related settings.)

Procedure: To configure presentation of the calling party number to the analog device connected to the FXS port associated with the interface

Mode: Interface FXS

Step	Command	Purpose
1	<code>node(if-fxs)[if-name]#[no] caller-id-presentation {pre-ring mid-ring}</code>	Enables/Disables presentation of the caller ID, and configures the time when caller ID is presented on the line.

Example: Enable the caller-ID presentation after the first ring

The following example shows how to enable caller-ID

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface fxs MyFxsIf
SN(if-fxs)[myFxsIf]#caller-id-presentation mid-ring
```

Configuring call holding supplementary service (optional)

The *call holding* supplementary service can be administratively enabled or disabled. If disabled, the user doesn't have the possibility to have two calls at the same time on an FXS interface.

Procedure: To configure call holding

Mode: Interface FXS

Step	Command	Purpose
1	<code>node(if-fxs)[if-name]#[no] call-hold</code>	Enables/Disables call holding supplementary service (Default: enabled)

Example: Disable call holding

The following example shows how to disable call holding

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface fxs MyFxsIf
SN(if-fxs)[myFxsIf]#no call-hold
```

Configuring call waiting supplementary service (optional)

The *call waiting* supplementary service can be administratively enabled or disabled. If disabled, the terminal is considered busy when in a call. An incoming call is thus not presented to the user, and the caller hears busy-tone.

The user of the device connected to the FXS port can be given the possibility to activate/deactivate call waiting by means of a special digit sequence touched on the keypad of his device (see section “[Configuring the precall service tables](#)” on page 322 for more information). The configuration in the FXS interface is administrative, this means if call waiting is disabled here, the user cannot activate it anymore.

Procedure: To configure call waiting

Mode: Interface FXS

Step	Command	Purpose
1	<code>node(if-fxs)[if-name]#[no] call-waiting</code>	Enables/Disables call waiting supplementary service (Default: enabled)

Example: Disable call waiting

The following example shows how to disable call waiting

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface fxs MyFxsIf
SN(if-fxs)[myFxsIf]#no call-waiting
```

Configuring additional call offering supplementary service (optional)

The *additional call offering* supplementary service can be administratively enabled or disabled. If disabled, the user doesn't have the possibility to make a callback during an active call.

Procedure: To configure additional call offering

Mode: Interface FXS

Step	Command	Purpose
1	<code>node(if-fxs)[if-name]#[no] additional-call-offering</code>	Enables/Disables additional call offering supplementary service (Default: enabled)

Example: Disable additional call offering

The following example shows how to disable call holding

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface fxs MyFxsIf
SN(if-fxs)[myFxsIf]#no additional-call-offering
```


Chapter 30 **FXO interface configuration**

Chapter contents

Introduction	336
FXO services description	337
Creating an FXO interface.....	337
Deleting an FXO interface.....	338
FXO interface configuration task list	339
Configuring an alternate PSTN profile (optional)	339
Configuring when the digits are dialed (optional)	340
Configuring the number of rings to wait before answering the call (optional)	341
Configuring how to detect a call has disconnected (optional)	342
Configuring how to detect an outgoing call is connected (optional)	343
Configuring the destination of the call	344
FXO interface examples	345

Introduction

This chapter provides an overview of FXO interfaces and the tasks involved in configuring them. This chapter does not explain the basic configuration steps common to all Context Switch (CS) interfaces. Information about basic interface configuration can be found in chapter 27, “[CS interface configuration](#)” on page 315.

An FXO, *Foreign eXchange Office*, interface connects to an FXS, *Foreign eXchange Subscriber*, interface. These two interfaces are used in analog telephony. The FXS interface is provided at the central office in order to connect to telephones, modems, PBXs, faxes, etc. Telephones and modems are FXO interfaces and want to connect to the central office. The FXO interface in the SmartNode products is like the telephone and modem interface.

In SmartWare, an FXO interface functions to connect the analog FXO port’s call signaling to the call control process in SmartWare. Recall that an interface in SmartNode products is a logical device and a port is a physical device. So the FXO feature consists of the logical interface with all its processes together with its configurable parameters and the physical interface for the actual analog, 2-wire connection to an FXS device. There is a one-to-one correspondence between the port and the interface.

In order for the interface to be able to make a connection over the 2-wire analog line, there must be a port bound to the interface (see [figure 59](#)). For more information on ports, interfaces, and binding, see section “[Interfaces, Ports, and Bindings](#)” on page 38. For specific details on binding the FXO port to an FXO interface, see section “[Bind FXO ports to higher layer applications](#)” on page 449.

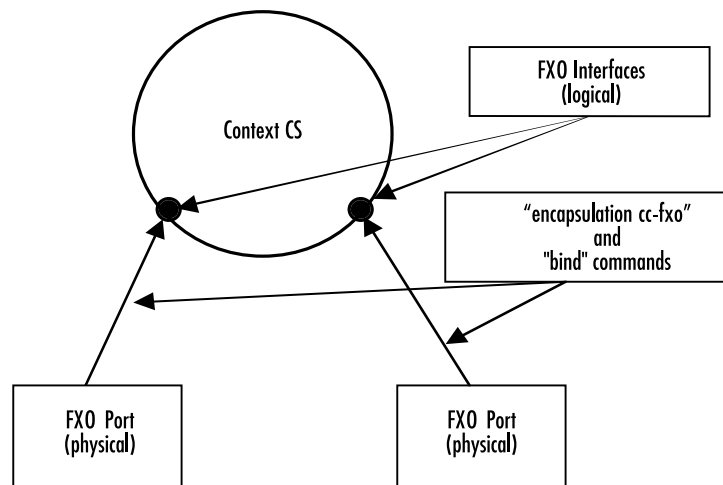


Figure 59. FXO interfaces on the CS context

This chapter includes the following sections:

- FXO services description (see [page 337](#))
- Creating an FXO interface (see [page 337](#))
- Deleting an FXO interface (see [page 338](#))
- FXO interface configuration task list (see [page 339](#))

FXO services description

The wide variety of applications and services are supported through a rich feature set. The major characteristics and features are

- 2-wire loop-start
- Off-hook and ring detection supervision
- Automatic and programmable line gain
- Programmable ring count before call pick-up
- End-of-call detection by line drop (call release indication), busy tone detection and battery reversal detection
- Hook-flash sending: programmable duration, H.245 hook-flash relay (“!” in user input) which provides Cisco compatibility
- DTMF send and detect: programmable interdigit timer, DTMF-relay
- Caller ID (CLID) FSK command line interface reception and relay to VoIP signaling (Bellcore/ANSI and ETSI/ITU)
- Call routing based on caller ID
- Second dial-tone for two-stage DTMF dialing with call routing based on DTMF numbers.

Creating an FXO interface

Interface names can be any arbitrary string. Use self-explanatory names for your interfaces to reflect their usage in your application. After creating the FXO interface, it is necessary to bind the FXO interface. Refer to chapter 37, “FXO port configuration” on page 447 for details.

Mode: Context CS

You enter Context CS, one of the configuration modes, as follows.

Note *Node* is the host name you have assigned to your SmartNode and is the basic prompt.

Step	Prompt & command	Purpose
1	node>	Basic prompt in Operator Exec mode
2	node>enable	Enters Administration execution mode
	node#	The prompt in administration execution mode
3	node#configure	Enters the Configure configuration mode
	node(cfg)#	The prompt in the Configure configuration mode
4	node(cfg)#context cs	Enters the context “CS” for Circuit Switch
	node(ctx-cs)[switch]#	The prompt in the Context CS configuration mode

Once you are in the Context CS mode, you can enter the FXO configuration mode with the next steps.

Step	Prompt & command	Purpose
5	node(ctx-cs)[switch]#interface fxo <i>name</i>	The "interface fxo" command creates the new interface name, which represents an FXO interface. This command also places you in the FXO interface configuration mode for the created interface.
6	node(if-fxo)[name]#	You are now in the FXO interface configuration mode. In this mode, you may configure the parameters for the FXO interface name

Example: Create an FXO interface named *PSTN-FALLBACK*

The following commands would be used—in Context CS mode—to create an FXO interface named *PSTN-FALLBACK*:

```
node(ctx-cs)[switch]#interface fxo PSTN-FALLBACK
node(if-fxo)[PSTN-FA~]#
```

Deleting an FXO 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. The **no** form of the FXO interface deletes the interface.

Mode: Context CS.

Step	Prompt & command	Purpose
1	node>	Basic prompt in Operator Exec mode
2	node>enable	Enters Administration execution mode
	node#	The prompt in administration execution mode
3	node#configure	Enters the Configure configuration mode
	node(cfg)#	The prompt in the Configure configuration mode
4	node(cfg)#context cs	Enters the context "CS" for Circuit Switch
	node(ctx-cs)[switch]#	The prompt in the Context CS configuration mode
5	node(ctx-cs)[switch]#no interface fxo <i>name</i>	Deletes the existing interface <i>name</i>

FXO interface configuration task list

There are numerous configurable parameters that apply to the FXO interface. The basic commands are listed with a short description of their function.

- **use profile pstn**—Defining and applying an alternate PSTN profile for a specific FXO interface
- **dial-after**—Selecting whether the FXO interface dials after a pre-defined time or after detection of dial-tone
- **ring-number**—Defining how many rings are received before answering an incoming call
- **disconnect-signal**—Selecting the method of determining a call has been disconnected
- **connect-signal**—Choosing how to detect the connection on the remote end of an outgoing call
- **route**—Determining the destination (interface) for the incoming call

Configuring an alternate PSTN profile (optional)

The PSTN profile has the following configurable parameters:

- Echo canceller (can be enabled or disabled)
- Output gain, which sets the volume of the output of the PSTN interface and port, in this case, FXO.

To define an alternate PSTN profile for the FXO interface, first create the profile according to instructions in chapter 41, “PSTN profile configuration” on page 493. Then you can apply the newly defined PSTN profile to a specific FXO interface with the **use** command as follows.

First enter the Interface FXO configuration mode.

Mode: Interface FXO

Step	Prompt, command & response	Purpose
1	node>	Basic prompt in Operator Exec mode
2	node>enable	Enters Administration execution mode
	node#	Response: The prompt in administration execution mode is the #
3	node#configure	Enters the Configure configuration mode
	node(cfg)#	Response: The prompt in the Configure configuration mode is (cfg)#
4	node(cfg)#context cs	Enters the Context CS configuration mode
	node(ctx-cs)[switch]#	Response: The prompt in the Context CS configuration mode is (ctx-cs)[switch]#
5	node(ctx-cs)[switch]#interface fxo if-name	Enter the Interface FXO configuration mode
	node(if-fxo)[if-name]#	Response: The prompt in the Interface FXO configuration mode is (if-fxo)[if-name]#

Now we can apply the PSTN profile for the FXO interface named name as follows.

Step	Prompt, command & response	Purpose
6	node(if-fxo)[if-name]#[no] use profile pstn profile-name	The "profile pstn" command is applied to the FXO interface named <i>if-name</i>
	node(if-fxo)[if-name]#	Response: You are now in the FXO interface configuration mode. In this mode, you may configure the parameters for the FXO interface <i>name</i> .

Configuring when the digits are dialed (optional)

When the FXO port goes off-hook to make an outgoing call, the FXS switch normally sends a dial-tone to indicate it is ready to received dialed digits. Alternatively, you can specify the FXO interface to wait a specific period of time after going off-hook before dialing the first digit.

Note All countries do not have the same dial-tone. For information on configuring the dial-tone for your country, refer to chapter 34, "Tone configuration" on page 417.

The default setting is to wait for the dial-tone. Choosing to wait a specific time after dial-tone, the variable is timeout, the number of seconds to wait. Zero (0) seconds means that the interface dials immediately.

Mode: Interface FXO

Step	Prompt, command & response	Purpose
1	node>	Basic prompt in Operator Exec mode
2	node>enable	Enters Administration execution mode
	node#	Response: The prompt in administration execution mode is the #
3	node#configure	Enters the Configure configuration mode
	node(cfg)#	Response: The prompt in the Configure configuration mode is (cfg)#
4	node(cfg)#context cs	Enters the Context CS configuration mode
	node(ctx-cs)[switch]#	Response: The prompt in the Context CS configuration mode is (ctx-cs)[switch]#

Step	Prompt, command & response	Purpose
5	node(ctx-cs)[switch]#interface fxo if-name	Enter the Interface FXO configuration mode
	node(if-fxo)[if-name]#	Response: The prompt in the Interface FXO configuration mode is (if-fxo)[if-name]#
6	node(if-fxo)[if-name]#dial-after {dial-tone timeout seconds}	Specifies whether to dial after detection of dial-tone (default) or to wait for a specified timeout (in seconds). Zero (0) seconds will initiate dialing immediately after going off-hook.

Example: Setting the timeout to be 4 seconds after going off-hook when initiating a call. The timeout is set for the FXO interface named *Line0*.

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface fxo Line0
node(if-fxo)[Line0]#dial-after timeout 4
```

You can verify the change in configuration by using the **show running-config** command.

This is one of the few commands that does not have a **no** inverse operation of the command. If you want to change the timeout period, re-enter the command with the new timeout period. The other option is to wait for dial-tone. To return to the default (waiting for dial-tone before dialing), enter the command again using **dial-tone**.

Note Verify that you have configured the dial-tone for the country in which the SmartNode is installed. (see chapter 34, “[Tone configuration](#)” on page 417). If the dial-tone is not configured for the proper country, the FXO interface will not detect when the remote FXS switch is sending dial-tone.

Configuring the number of rings to wait before answering the call (optional)

An FXO port identifies an incoming call by detecting the ring from the FXS switch. The **ring-number** is a configurable parameter which selects the number of rings before answering the incoming call, that is, before going off-hook and establishing the call. The minimum value for **ring-number** is zero (0). With a ring-number of zero, the FXO interface never answers an incoming call.

Due to variations between countries, the proper setting may be 1 or 2. In the USA the Caller-ID (CLID) is sent to the FXO port between the first and second ring, so a ring-number of 2 would be appropriate. On the other hand, numerous countries send the CLID prior to the first ring, so the default setting of 1 would be satisfactory.

Mode: Interface FXO

Step	Prompt, command & response	Purpose
1	node>	Basic prompt in Operator Exec mode
2	node>enable	Enters Administration execution mode
	node#	Response: The prompt in administration execution mode is the #
3	node#configure	Enters the Configure configuration mode
	node(cfg)#	Response: The prompt in the Configure configuration mode is (cfg)#
4	node(cfg)#context cs	Enters the Context CS configure mode
	node(ctx-cs)[switch]#	Response: The prompt in the Context CS configuration mode is (ctx-cs)[switch]#
5	node(ctx-cs)[switch]#interface fxo if-name	Enter the Interface FXO configuration mode
	node(if-fxo)[if-name]#	Response: The prompt in the Interface FXO configuration mode is (if-fxo)[if-name]#
6	node(if-fxo)[if-name]#ring-number count	Specifies the number of rings to wait before going off-hook. Default = 1 ring.

Example: Configure the ring number to wait for CLID by setting the *count* to 2. The name of the specific FXO interface is *pstn-local*.

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface fxo pstn-local
node(if-fxo)[pstn-local]#ring-number 2
```

You can verify the change in configuration by using the **show running-config** command.

Configuring how to detect a call has disconnected (optional)

When a call has disconnected, the FXO interface may detect and verify the termination of the phone call by three different methods.

- Detect a busy tone or release tone
- Detect a loop break (if provided by the FXS switch)
- Detect battery reversal (if provided by the FXS switch)

The selection of the method, if any of the three, is via the **disconnect-signal** command. The default enables only loop-break. Upon detecting a *loop break*, the FXO interface proceeds to clear the call on the SmartNode. In some instances, the user may need to transmit all of the in-band information (tone signal, announcement) to the end party after disconnection has occurred.

The **disconnect-signal** command can be used to enable or disable the three detection methods. If all three methods are disabled, the call is cleared after a period of 30 seconds from reception of the disconnect signal. Consequently it becomes the responsibility of the end party to execute the on-hook (completing the disconnection phase) for the call to be completely cleared.

Note Verify that the busy and release tones are correctly configured for the country where the SmartNode is installed (see chapter 34, “Tone configuration” on page 417 for configuration information. If the tones are improperly configured, the FXO port will not detect them, resulting in missed phone calls.

Mode: Interface FXO

Step	Prompt, command & response	Purpose
1	node>	Basic prompt in Operator Exec mode
2	node>enable	Enters Administration execution mode
	node#	Response: The prompt in administration execution mode is the #
3	node#configure	Enters the Configure configuration mode
	node(cfg)#	Response: The prompt in the Configure configuration mode is (cfg)#
4	node(cfg)#context cs	Enters the Context CS configure mode
	node(ctx-cs)[switch]#	Response: The prompt in the Context CS configuration mode is (ctx-cs)[switch]#
5	node(ctx-cs)[switch]#interface fxo if-name	Enter the Interface FXO configuration mode
	node(if-fxo)[if-name]#	Response: The prompt in the Interface FXO configuration mode is (if-fxo)[if-name]#
6	node(if-fxo)[if-name]#[no] disconnect-signal {battery-reversal busy-tone loop-break}	The default is Loop-break. To disable it, use the no inverse command. Should all three methods be disabled, the call is cleared 30 seconds after receiving the disconnect signal. The default setting of loop-break is not displayed in the running-config output.

Configuring how to detect an outgoing call is connected (optional)

An FXO interface has the following methods for verifying the connection of an outgoing call after the dialing has been completed:

- Detect battery reversal (if provided by the FXS switch)
- Detect the first tax pulse (if provided by the FXS switch)

Note **Tax Impulse Signals:** European telephone companies in Austria, Belgium, Czechoslovakia, Germany, Spain and Switzerland place a *pulse signal* on the phone line to meter the length of the telephone call for billing purposes.

The command to enable or disable these methods is **connect-signal**. If both are enabled, only one needs to occur for the FXO interface to verify a properly connected call with the remote party. Should both be disabled, the SmartNode waits for the call-connect signal from the FXS switch.

Mode: Interface FXO

Step	Prompt, command & response	Purpose
1	node>	Basic prompt in Operator Exec mode
2	node>enable	Enters Administration execution mode
	node#	Response: The prompt in administration execution mode is the #
3	node#configure	Enters the Configure configuration mode
	node(cfg)#	Response: The prompt in the Configure configuration mode is (cfg)#
4	node(cfg)#context cs	Enters the Context CS configure mode
	node(ctx-cs)[switch]#	Response: The prompt in the Context CS configuration mode is (ctx-cs)[switch]#
5	node(ctx-cs)[switch]#interface fxo if-name	Enter the Interface FXO configuration mode
	node(if-fxo)[if-name]#	Response: The prompt in the Interface FXO configuration mode is (if-fxo)[if-name]#
6	node(if-fxo)[if-name]#[no] connect-signal {battery-reversal tax-pulse}	Selects battery-reversal, tax-pulse or neither to determine when outgoing calls are connected. Default: both methods are disabled.

Note Only disable connect-signal if you are sure that the FXS switch provides a call connect signal.

Configuring the destination of the call

The last command in configuring the FXO Interface is the route command. This command configures the call router. You can configure the routing-destination for call setup and for service activation. For complete details, see chapter 33, “Call router configuration” on page 361.

Mode: Interface FXO

Step	Prompt, command & response	Purpose
1	node>	Basic prompt in Operator Exec mode
2	node>enable	Enters Administration execution mode
	node#	Response: The prompt in administration execution mode is the #
3	node#configure	Enters the Configure configuration mode
	node(cfg)#	Response: The prompt in the Configure configuration mode is (cfg)#
4	node(cfg)#context cs	Enters the Context CS configure mode
	node(ctx-cs)[switch]#	Response: The prompt in the Context CS configuration mode is (ctx-cs)[switch]#

Step	Prompt, command & response	Purpose
5	node(ctx-cs)[switch]#interface fxo <i>if-name</i>	Enter the Interface FXO configuration mode
	node(if-fxo)[if-name]#	Response: The prompt in the Interface FXO configuration mode is (if-fxo) [if-name]#
6	node(if-fxo)[if-name]#[no] route {call {dest-interface <i>interface-name</i> dest-service <i>table-name</i> dest-table <i>service-name</i> } precall {dest-interface <i>interface-name</i> dest-service <i>table-name</i> dest-table <i>service-name</i> }	Use this command to route a call (dest-interface) directly to an interface specified with the <i>interface-name</i> parameter, (dest-table) to the call router using the <i>table-name</i> table as the first routing table, or (dest-table) directly to a service specified with the <i>service-name</i> parameter.

FXO interface examples

Example 1: Configuring an FXO interface which is to be connected to a PSTN network for analog line extension over IP. The FXS switch provides caller-id between the first and second ring and uses battery reversal to indicate a connected call. The FXO interface is named *pstn-local*. The incoming call is routed directly to the interface named *pstn-1-voip*.

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface fxo pstn-local
node(if-fxo)[pstn-local]#connect-signal battery-reversal
node(if-fxo)[pstn-local]#ring-number 2
node(if-fxo)[pstn-local]#route call dest-interface pstn-1-voip
```

Example 2: Configuring an FXO interface to be used as fallback if the IP network link is down. This means that there are only out-going calls. You are not sure whether the FXS switch provides a connect signal. In this case, you only have to create the interface and bind the FXO port to the FXO interface. (For binding the FXO port to the FXO interface, see chapter 37, “FXO port configuration” on page 447.)

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface fxo pstn-fb
node(if-fxo)[pstn-fb]#connect-signal battery-reversal
node(if-fxo)[pstn-fb]#connect-signal tax-pulse
```


Chapter 31 **H.323 interface configuration**

Chapter contents

Introduction	348
H.323 interface configuration task list	348
Binding the interface to an H.323 gateway	349
Configuring an alternate VoIP profile (optional)	350
Configuring CLIP/CLIR support (optional)	351
Enabling the early call disconnect (optional)	352
Enabling the via address support (optional)	353
Override the default destination call signaling port (Optional)	353
Configuring status inquiry settings (optional)	354

Introduction

This chapter provides an overview of H.323 interfaces used by H.323 gateways and describes the specific tasks involved in their configuration. This chapter does not explain the basic configuration steps required for all CS interfaces. Information about basic interface configuration can be found in the general chapter about CS interface configuration.

Within the CS context of SmartWare, an H.323 interface is a special type of CS interface providing call routing for incoming and outgoing calls to and from the H.323 gateway (see [figure 60](#)).

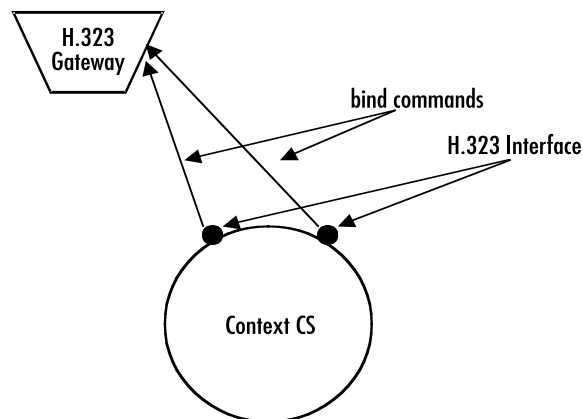


Figure 60. H.323 interfaces on the CS context

An H.323 interface is a CS interface type that also provides voice over IP settings in addition to the general CS interface parameters. All H.323 interfaces must be explicitly bound to an H.323 gateway. Calls, which are routed from the Context CS to one of the H.323 interfaces, will be forwarded for call establishment to the H.323 gateway to which the H.323 interface is bound. All the parameters configured in the H.323 interface will be applied to the forwarded call.

When a call arrives over H.323 in the H.323 gateway. The gateway looks for the best matching H.323 interface, which is bound to that gateway. If there is an H.323 interface, which contains the IP address of the source of the H.323 call in its *remoteip* configuration parameter, the call will be handed over to that interface for further call processing. If no such interface is found, the gateway looks for an interface, which is bound to that gateway and does not contain a *remoteip* parameter. If such an interface is found the call will be handed over to that interface for further processing. If however such an interface is also not available, the call will be dropped.

H.323 interface configuration task list

This section describes the configuration tasks for H.323 interfaces listed below. You must at least perform the tasks, which are not marked as optional, to define a working H.323 interface. The optional tasks are usually only required in advanced configurations. Before you can start with the H.323 interface specific configuration tasks, you need to create the H323 interface and define the routing for it as defined in chapter 27, “[CS interface configuration](#)” on page 315.

- Binding the interface to an H.323 gateway
- Configuring a remote IP address
- Using an alternate VoIP profile (optional)

- Configuring information transfer capability handling (optional)
- Configuring CLIP/CLIR support (optional)
- Enabling the early call disconnect (optional)
- Enabling the via address support (optional)
- Overriding the default destination call-signaling port (optional)
- Configuring status inquiry settings (optional)

Binding the interface to an H.323 gateway

Every H.323 interface must be explicitly bound to an H.323 gateway instance.

Procedure: To bind an H.323 interface to an H.323 gateway.

Mode: Interface H.323

Step	Command	Purpose
1	node(if-h323)[if-name]#bind gateway gw-name	Binds the gateway to an H.323 gateway.

Examples: Bind the H.323 interfaces to an H.323 gateway instance

The following example shows how to bind an H.323 interface named *MyH323If* to an H.323 gateway instance named *h323*.

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface h323 MyH323If
SN(if-h323)[MyH323If]#bind gateway h323
Configure a remote IP address
```

If the gateway to which the H.323 interface is bound does not use a gatekeeper, it is required to specify the IP address of the remote entity for which the H.323 interface is used directly within the H.323 interface. This is done using the procedure below. If the H.323 gateway however uses a gatekeeper, the gatekeeper is responsible for resolving the remote entities IP address. In that case this procedure must not be used.

Procedure: To specify the remote H.323 entities IP address in the H.323 interface.

Mode: Interface H.323

Step	Command	Purpose
1	node(if-h323)[if-name]#remoteip ip-address	Defines the IP address of the remote H.323 entity, for which this interface shall be used.

Examples: Define the IP address of the remote H.323 entity

The following example shows how to associate an H.323 interface named *MyH323If* with a remote H.323 entity, which has the IP address 1.2.3.4

```
SN>enable
```

```
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface h323 MyH323If
SN(if-h323)[myh323if]#remoteip 1.2.3.4
```

Configuring an alternate VoIP profile (optional)

Normally, the VoIP profile defined in the H.323 gateway is used for all the calls over that gateway. However, it is possible to specify an alternate VoIP profile in the H.323 interface. In that case the VoIP profile defined within the VoIP interface is used for all the calls established using that H.323 interface instead of the VoIP profile defined in the H.323 gateway.

Procedure: To define an alternate VoIP profile for the H.323 interface

Mode: Interface H.323

Step	Command	Purpose
1	<code>node(if-h323)[if-name]#use profile voip profile-name</code>	Defines an alternate VoIP profile to be used for this VoIP interface

Example: Configure an alternate VoIP profile

The following example shows how to replace the default VoIP profile of the H.323 gateway with a VoIP profile named *myprofile* for an H.323 interface named *MyH323If*.

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface h323 MyH323If
SN(if-h323)[MyH323If]#use profile voip myprofile
Configure information transfer capability handling (Optional)
```

Normally, the H.323 gateway transparently forwards the information transfer capability information element between the H.323 network and other gateways of the SmartNode. There are, however, several H.323 clients that do not provide correct information transfer capability information. One of the most often used clients of this type is Microsoft Netmeeting. When communicating to one of these clients, it is necessary to define the correct information transfer capability in the H.323 interface. It is possible to define for each direction (for calls from or to H.323) separately, whether the information transfer capability received from the network, or another information transfer capability should be used for the calls.

Note The default behavior when not configured otherwise is to set the information transfer capability of incoming calls to 3k1-audio and to transparently pass the information transfer capability for outgoing calls.

Procedure: To configure information transfer capability overriding

Mode: Interface H.323

Step	Command	Purpose
1	<code>node(if- h323)[if-name]#itc rx {3k1-audio 7k-audio restricted-digital unrestricted-digital speech video transparent }</code>	Specifies the information transfer capability to be used for calls from the H.323 gateway to another gateway of the system (incoming calls). All settings force the specified information transfer capability to be used except for the transparent setting, which indicates that the information transfer capability of the call should be forwarded transparently.
2	<code>node(if- h323)[if-name]#itc tx {3k1-audio 7k-audio restricted-digital unrestricted-digital speech video transparent }</code>	Specifies the information transfer capability to be used for calls from any gateway of the system to the H.323 gateway (outgoing calls). All settings force the specified information transfer capability to be used except for the transparent setting, which indicates that the information transfer capability of the call should be forwarded transparently.

Example: Configure information transfer capability handling

In the following example the information transfer capability for inbound calls through the H.323 interface *Myh323If* is forced to speech. This is an appropriate setting, when communicating to Microsoft Netmeeting clients.

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface h323 MyH323If
SN(if-h323)[MyH323If]#itc rx speech
```

Configuring CLIP/CLIR support (optional)

According to the H.323 standard, information about *calling line identification presentation/calling line identification restriction* (CLIP/CLIR) is not provided, when using the H.323 protocol. However, there are H.323 equipment vendors, which allow tunneling this information through an H.323 connection. The additional information is inserted in octet 3a of the calling party number information element in the Q.931 part of the H.323 setup message.

Note This functionality is not standardized and might cause interoperability problems, if enabled.

Procedure: To enable tunnelling of CLIP/CLIR information over H.323

Mode: Interface H.323

Step	Command	Purpose
1	<code>node(if- h323)[if-name]#clip-clir-support</code>	Enables CLIP/CLIR support on the H.323 interface

Example: Enable CLIP/CLIR support

The following example shows how to enable CLIP/CLIR support on the H.323 interface MyH323If.

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface h323 MyH323If
SN(if-h323)[MyH323If]#clip-clir-support
```

Enabling the early call disconnect (optional)

Early call disconnect suppresses busy tones (e.g. disturbing a telephone conference) and post-call announcements by sending an H.323 Release message to the remote peer 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).

Procedure: To enable early call disconnect

Mode: Interface H.323

Step	Command	Purpose
1	<code>node(if-h323)[if-name]#early-disconnect</code>	Enables early call disconnect (Default: disabled)

Example: Enable early call disconnect

The following example shows how to enable early call disconnect on an H.323 interface named MyH323If.

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface h323 MyH323If
SN(if-h323)[MyH323If]#early-disconnect
```

Enabling the via address support (optional)

Some LAN Voice applications require the H.323 gateway to add the calling party number of the connected terminal as an H.323 E.164 Alias to the Facility message when transferring a call to another gateway. This enables a gatekeeper to detect loops of call forwarding and to stop them.

Procedure: To enable sending of the via address in call transfers

Mode: Interface H.323

Step	Command	Purpose
1	node(if-h323)[if-name]#via-address-support	Enables sending of the via address in call transfers (Default: disabled)

Example: Enabling the via address support

The following example shows how to enable the via address support on an H.323 interface named MyH323If.

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface h323 MyH323If
SN(if-h323)[MyH323If]#via-address-support
```

Override the default destination call signaling port (Optional)

Normally, if no gatekeeper is used, the TCP call-signaling connection for outbound H.323 calls is established to the H.323 standard call-signaling port 1720. If your destination uses a different call-signaling port, it is possible to define an alternate port using this procedure.

Note The call-signaling port specified here has no effect, if a gatekeeper is used. In that case the gatekeeper will provide the portnumber to use for establishing the call signaling connection

Procedure: To configure an alternate destination TCP call-signaling port

Mode: Interface H.323

Step	Command	Purpose
1	node(if-h323)[if-name]# remoteport port	Specifies the TCP port to which the call-signaling connection should be established.

Example: Specifying an alternate destination call-signaling port

The following example shows how to set the destination call-signaling port number for the H.323 interface *MyH323If* to 2300.

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface h323 MyH323If
SN(if-h323)[MyH323If]#remoteport 2300
```

Configuring status inquiry settings (optional)

Normally, the H.323 gateway will send out status inquiries every minute on each connected H.323 call. According to the H.323 standard, the remote entity must respond to these status inquiries, which allows the H.323 gateway to detect, if the call on the remote H.323 entity is still alive. If no response is received after another minute, the call will be dropped.

Unfortunately, there are H.323 entities, which do not respond to these status inquiries. This causes every call to be dropped after being connected for two minutes using the default setting.

As a workaround for these non-compliant implementations, it is possible to disable the status inquiry checking.

It is also possible to change the default status inquiry interval of 60 seconds to a different value, if required.

Procedure: To disable status inquiries

Mode: Interface H.323

Step	Command	Purpose
1	<code>node(if-h323)[if-name]#no status-inquiry</code>	Disables status inquiries on the interface

Example: Disable status inquiries

The following example shows how to disable status inquiries for calls handled by the H.323 interface *MyH323If*:

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface h323 MyH323If
SN(if-h323)[MyH323If]#no status-inquiry
```

Procedure: To change the default status inquiry interval

Mode: Interface H.323

Step	Command	Purpose
1	<code>node(if-h323)[if-name]#status-inquiry timeout seconds</code>	Changes the status inquiry interval on the interface to the specified number of seconds

Example: Disable status inquiries

The following example shows how to change the status inquiry interval for the H.323 interface *MyH323If* to 120 seconds.

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface h323 MyH323If
SN(if-h323)[MyH323If]#status-inquiry timeout 120
```

Chapter 32 **SIP interface configuration**

Chapter contents

Introduction	356
SIP interface configuration task list.....	356
Binding the interface to a SIP gateway	357
Configure a remote host	357
Configuring an alternate VoIP profile (Optional)	358
Configuring early call connect / disconnect (optional)	358
Configuring a phone context (optional)	359

Introduction

This chapter provides an overview of SIP interfaces used by 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. Information about basic interface configuration can be found in the general chapter about CS interface configuration.

Within the CS context of SmartWare, a SIP interface is a special type of CS interface providing call routing for incoming and outgoing calls to and from the SIP gateway (see [figure 61](#)).

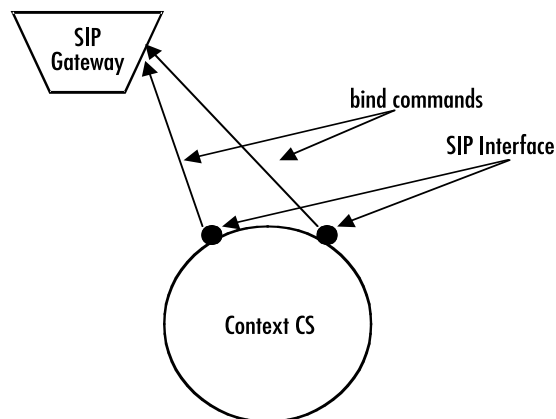


Figure 61. SIP interfaces on the CS context

A SIP interface is a CS interface type that also provides voice over IP settings in addition to the general CS interface parameters. All SIP interfaces must be explicitly bound to a SIP gateway. Calls, which are routed from the Context CS to one of the SIP interfaces, will be forwarded for call establishment to the SIP gateway to which the SIP interface is bound. All the parameters configured in the SIP interface will be applied to the forwarded call.

When a call arrives over SIP in the SIP gateway. The gateway looks for the best matching SIP interface, which is bound to that gateway. If there is a SIP interface, which contains the IP address of the source of the SIP call in its remote configuration parameter, the call will be handed over to that interface for further call processing. If no such interface is found, the gateway looks for an interface, which is bound to that gateway and does not contain a remote parameter. If such an interface is found the call will be handed over to that interface for further processing. If however such an interface is also not available, the call will be dropped.

SIP interface configuration task list

This section describes the configuration tasks for SIP interfaces listed below. You must at least perform the tasks, which are not marked as optional, to define a working SIP interface. The optional tasks are usually 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 27, “CS interface configuration”](#) on page 315.

- Binding the interface to a SIP gateway
- Configure a remote SIP URI
- Using an alternate VoIP profile (Optional)

- Configure early call connect / disconnect (Optional)
- Configure a phone context (Optional)

Binding the interface to a SIP gateway

Every SIP interface must be explicitly bound to a SIP gateway instance.

Procedure: To bind a SIP interface to a SIP gateway.

Mode: Interface SIP

Step	Command	Purpose
1	node(if-sip)[if-name]#bind gateway gw-name	Binds the gateway to a SIP gateway.

Examples: Bind the SIP interfaces to a SIP gateway instance

The following example shows how to bind a SIP interface named MySipIf to a SIP gateway instance named sip.

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface sip MySipIf
SN(if-sip)[MySipIf]#bind gateway sip
```

Configure a remote host

You can define a remote host within the SIP interface. All calls forwarded to the SIP gateway through this SIP interface will be sent to that host unless the SIP gateway has a default server configured. In that case the remote host specified here will only be used for the request-URI and the to-header, but the request will be sent to the default server.

The remote host parameter may contain either an IP address or a DNS hostname.

Usually, you only need to set this parameter, if you do not have a default server configured on the SIP gateway. This is the case, if you either do not use a SIP server for call routing, or if the server you use for call routing is a SIP redirect server.

Procedure: To specify the remote SIP entities IP address or DNS hostname in the SIP interface

Mode: Interface SIP

Step	Command	Purpose
1	node(if-sip)[if-name]#remoteip ip-address	Defines the IP address or DNS hostname of the remote SIP entity, for which this interface shall be used.

Examples: Define the remote SIP entity using an IP address

The following example shows how to associate a SIP interface named MySipIf with a remote SIP entity, which has the IP address 1.2.3.4

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface sip MySipIf
SN(if-sip)[mySipIf]#remote 1.2.3.4
```

Examples: Define the remote SIP entity using a DNS hostname

The following example shows how to associate a SIP interface named MySipIf with a remote SIP entity, which has DNS hostname sipgw.mycompany.com

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface sip MySipIf
SN(if-sip)[mySipIf]#remote sipgw.mycompany.com
```

Configuring an alternate VoIP profile (Optional)

Normally, the VoIP profile defined in the SIP gateway is used for all the calls over that gateway. However, it is possible to specify an alternate VoIP profile in the SIP interface. In that case the VoIP profile defined within the VoIP interface is used for all the calls established using that SIP interface instead of the VoIP profile defined in the SIP gateway.

Procedure: To define an alternate VoIP profile for the SIP interface

Mode: Interface SIP

Step	Command	Purpose
1	<code>node(if-sip)[if-name]#use profile voip profile-name</code>	Defines an alternate VoIP profile to be used for this VoIP interface

Example: Configure an alternate VoIP profile

The following example shows how to replace the default VoIP profile of the SIP gateway by an alternate VoIP profile named myprofile for a SIP interface named MySipIf.

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface sip MySipIf
SN(if-sip)[MySipIf]#use profile voip myprofile
```

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. There are however several SIP user agents, which 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 in-band information is available. This will allow any SIP user agent to receive pre-call in-band 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).

Procedure: To enable early call connect and early call disconnect

Mode: Interface SIP

Step	Command	Purpose
1	<code>node(if-sip)[if-name]#early-connect</code>	Enables early call connect (Default: disabled)
2	<code>node(if-sip)[if-name]#early-disconnect</code>	Enables early call disconnect (Default: disabled)

Example: Enable early call connect and early call disconnect

The following example shows how to enable early call disconnect on a SIP interface named MySipIf.

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface sip MySipIf
SN(if-sip)[MySipIf]#early-connect
SN(if-sip)[MySipIf]#early-disconnect
```

Configuring a phone context (optional)

The SIP gateway automatically adds a *user=phone* tag to each SIP URI within INVITE requests, which contain a phone number in international format in their user part. Phone numbers in international format are numbers starting with a + sign. If the user part however contains a different type of phone number, the *user=phone* parameter will not be added automatically. This is because the *user=phone* option must not be appended to a SIP URI containing other than international numbers in their user part, unless a *phone-context* parameter is also appended to the SIP URI. Therefore, you can specify here the name of a phone context, to be included together with the *user=phone* parameter in any SIP URI sent within INVITE messages, which contain other than international format phone numbers in their user part.

Procedure: To define a phone context for the SIP interface

Mode: Interface SIP

Step	Command	Purpose
1	<code>node(if-sip)[if-name]#phone-context</code>	Specify a phone context for the interface

Example: Configure a phone-context

The following example shows how to configure a phone-context named mycompany for the SIP interface MySipIf.

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface sip MySipIf
SN(if-sip)[MySipIf]#phone-context mycompany
```

Using the configuration of this example, the URIs build by the SmartNode look as follows:

```
sip:+41318432343@mysipdomain.com;user=phone  
sip:531@mysipdomain.com;user=phone;phone-context=mycompany  
sipbob@mysipdomain.com
```

Without the special configuration of this example, these URIs would look as follows:

```
sip:+41318432343@mysipdomain.com;user=phone  
sip:531@mysipdomain.com  
sipbob@mysipdomain.com
```

Chapter 33 **Call router configuration**

Chapter contents

Introduction	363
Call router configuration task list.....	365
Map out the goals for the call router	365
Enable advanced call routing on circuit interfaces	366
Configure general call router behavior	366
Configure address completion timeout	366
Configure default digit collection timeout and terminating character	367
Configure number prefix for ISDN number types	368
Configure call routing tables	369
Create a routing table	370
Called party number routing table	372
Regular Expressions	372
Digit Collection	374
Digit Collection Variants	375
Calling party number routing table	378
Number type routing table	379
Numbering plan routing table	380
Name routing table	380
IP address routing table	381
URI routing table	381
Presentation Indicator Routing Table	381
Screening Indicator Routing Table	382
Information transfer capability routing table	383
Time of day routing table	384
Day of Week Routing Table	384
Date routing table	385
Deleting routing tables	385
Configure mapping tables	386
E.164 to E.164 Mapping Tables	391
Other mapping tables	393
Deleting mapping tables	394
Creating complex functions	395
Deleting complex functions	395
Creating call services	396
Creating a hunt group service	397
Creating a distribution group service	406
Deleting call services	408
Activate the call router configuration	408
Test the call router configuration	409

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 SmartWare. 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 criterias 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 62 illustrates direct call and advanced call routing. In this chapter, advanced call routing is explained. For configuring direct call routing refer to chapter 33, “Call router configuration” on page 361.

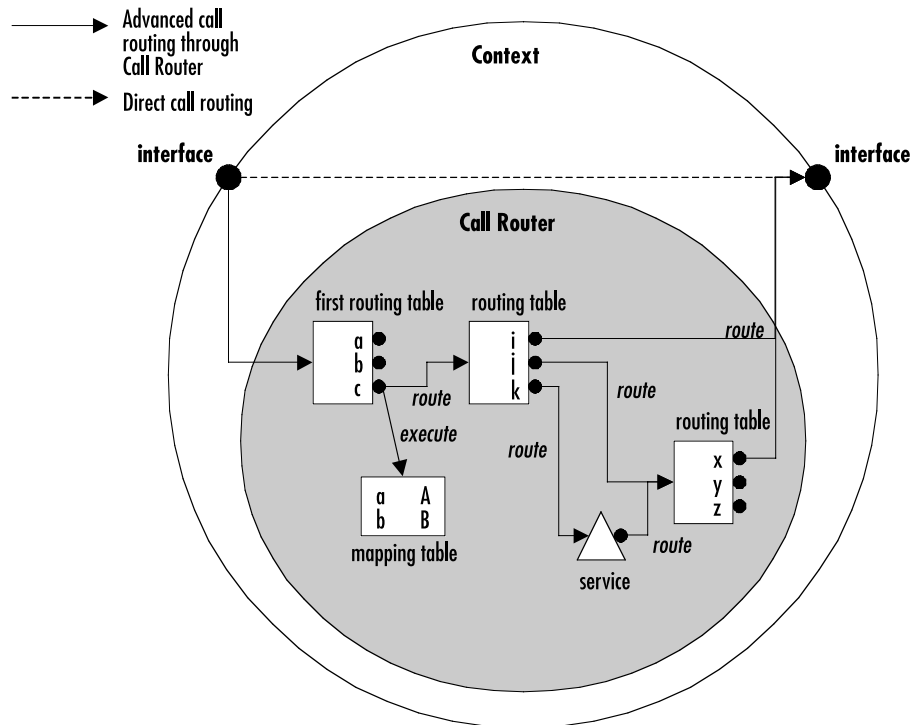


Figure 62. Direct call routing vs. advanced call routing

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 SmartWare 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 numbertype
- 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 out the goals for the call router

There are many possible policies and factors that may influence the call router configuration. Some examples are:

- Least cost routing
- 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 are:

- Available number ranges (DDI, MSN, PISN)

- Potential restrictions imposed by neighboring equipment (Gatekeepers, 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 62](#). 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.

Note Unlike previous versions of SmartWare, you cannot specify a fallback destination, which is used when the first destination fails. Instead route the call to a hunt-group service, which hunts the call over multiple destination interfaces as explained in section “[Creating a hunt group service](#)” on page 397.

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

```
SN[switch]#address-completion timeout 20
```

Configures the address completion timeout to 20s. 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 372). 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]#</code> digit-collection timeout <i>timeout</i>	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]#</code> digit-collection terminating-char <i>char</i> or no digit-collection terminating-char	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

```
SN[switch]#digit-collection timeout 3
SN[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 15 illustrates the three number types.

Table 15. 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 SmartWare 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 numbertype 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

```
SN[switch]#national-prefix 0041
```

```
Input: 99888xxxx Result: 004199888xxxx
```

```
SN[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 63. 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. The following call properties can be used as a routing table type:

Table 16. 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.
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.

Table 16. Routing table types (Continued)

Type	Description
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 to enable least cost routing.
date	Route calls based on the current date to enable least cost routing.
day-of-week	Route calls based on the current day of week to enable least cost routing.

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 <i>table-type table-name</i></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 <i>key</i> dest-interface <i>if-name</i> <i>function</i></code> or <code>node(rt-tab)[table-name]#route <i>key</i> dest-table <i>table-name</i> <i>function</i></code> or <code>node(rt-tab)[table-name]#route <i>key</i> dest-service <i>service-name</i> <i>function</i></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

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#routing-table called-e164 NATIONAL
SN(rt-tab)[NATIONAL]#route 001 dest-interface USVOIP-A
SN(rt-tab)[NATIONAL]#route 001320 dest-interface USVOIP-B
SN(rt-tab)[NATIONAL]#route 0044 dest-interface EUROVOIP
SN(rt-tab)[NATIONAL]#route 0049 dest-interface EUROVOIP
SN(rt-tab)[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 17. 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 nonconsecutive 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 18. Wildcard symbols used as keys in E.164 tables (calling-e164, called-e164)

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 or more times; for example, 88825, 888255, or 8882555555555
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:

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#routing-table called-e164 collect
SN(rt-tab)[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.

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#routing-table called-e164 DIST
SN(rt-tab)[DIST]#route 5.. dest-interface PBX
SN(rt-tab)[DIST]#route 001T dest-interface USVOIP
SN(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.

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#routing-table called-e164 COLLECT
SN(rt-tab)[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 than 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 before)	-	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.

Dialed Number	Selected Entry	Description
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.
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.

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#routing-table calling-e164 EXTS
SN(rt-tab)[EXTS]#route .%52[35]$ dest-interface breakout
SN(rt-tab)[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 specified 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.

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#routing-table calling-type-of-number TON
SN(rt-tab)[TON]#route international dest-table TAB-INCOMING-INT
SN(rt-tab)[TON]#route national dest-table TAB-INCOMING-NAT
SN(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.

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#routing-table calling-numbering-plan NP
SN(rt-tab)[NP]#route isdn-telephony dest-table TAB-INCOMING-ISDN
SN(rt-tab)[NP]#route data dest-table TAB-INCOMING-DATA
SN(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.

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#routing-table calling-name name
```

```
SN(rt-tab)[name]#route John Smith dest-table TAB-FROM-JOHN
SN(rt-tab)[name]#route Administrator dest-table TAB-FROM-ADMIN
SN(rt-tab)[name]#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, e.g. the IP address of the originating H.323 peer terminal. 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 and H.323 calls use the calling party IP address property to store the IP address of the remote SIP user agent or H.323 terminal, 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 H.323 terminal in the LAN to the next table TAB-FROM-LAN and all other calls to TAB-FROM-WAN.

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#routing-table calling-ip ip
SN(rt-tab)[ip]#route 172.16.32.0/24 dest-table TAB-FROM-LAN
SN(rt-tab)[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 an 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. Thus the called URI table is of limited use only.

Note Incoming SIP calls use the Calling Party URI Address property to store the From-URI of the remote SIP user agent. Other interfaces don't set the URI.

Example: Calling IP address routing table

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 route to the next table TAB-FROM-UNKNOWN.

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#routing-table calling-uri uri
SN(rt-tab)[ip]#route sip:john.smith@nil.net dest-table TAB-FROM-JOHN
SN(rt-tab)[ip]#route default dest-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. Incoming H.323 calls only set the presentation indicator transparently when Octet3a handling is enabled. Other 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.

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#routing-table calling-pi PI
SN(rt-tab)[PI]#route default dest-interface IF-OUT NO-CNPN
SN(rt-tab)[PI]#exit
SN(rt-tab)[switch]#mapping-table calling-pi to calling-e164 NO-CNPN
SN(map-tab)[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. Incoming H.323 calls only set the screening indicator transparently when Octet3a handling is enabled. 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.

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#routing-table calling-si SI
SN(rt-tab)[SI]#route user-failed none
SN(rt-tab)[SI]#route default dest-interface IF-OUT NO-CNPN
SN(rt-tab)[SPI]#exit
SN(rt-tab)[switch]#mapping-table calling-si to calling-e164 NO-CNPN
SN(map-tab)[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.

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#routing-table itc ITC
SN(rt-tab)[ITC]#route unrestricted-digital dest-interface IF-ACCESS
SN(rt-tab)[ITC]#route 7k-audio dest-interface IF-LOCAL-BREAKOUT
SN(rt-tab)[ITC]#route video dest-interface IF-ACCESS
SN(rt-tab)[ITC]#route default dest-interface IF-VOIP-CARRIER-A
```

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 following 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

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#routing-table time WORKDAY
SN(rt-tab)[WORKDAY]#route 08:00:00-16:59:59 dest-table TAB-BEST-QUAL
SN(rt-tab)[WORKDAY]#route 17:00:00-20:59:59 dest-interface IF-VOIP-A
SN(rt-tab)[WORKDAY]#route 21:00:00-23:59:59 dest-interface IF-VOIP-B
SN(rt-tab)[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 weekday routing table entries use the following commands starting in the CS context configuration mode.

Example: Day of week routing table

```

SN(cfg)#context cs
SN(ctx-cs)[switch]#routing-table day-of-week TAB-DAY
SN(rt-tab)[TAB-DAY]#route saturday dest-table TAB-LEAST-COST
SN(rt-tab)[TAB-DAY]#route sunday dest-table TAB-LEAST-COST
SN(rt-tab)[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 following format,

dd:mm:yyyy-dd:mm:yyyy

The full range must be specified.

Example: Date routing table

```

SN(cfg)#context cs
SN(ctx-cs)[switch]#routing-table day-of-week HOLIDAY2001
SN(rt-tab)[HOLIDAY~]#route 01.01.2001-02.01.2001 dest-table TAB-HOL
SN(rt-tab)[HOLIDAY~]#route 05.01.2001-05.01.2001 dest-table TAB-HOL
SN(rt-tab)[HOLIDAY~]#route 24.12.2001-31.12.2001 dest-table TAB-HOL
SN(rt-tab)[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 on 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 <i>table-name</i></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)[<i>table-name</i>]#no route <i>key</i></code>	Remove the entry with the specified key.
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:

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#routing-table MY-TABLE
SN(rt-tab)[MY-TABLE]#no route 10
SN(rt-tab)[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

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#no routing-table MY-TABLE
```

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	}	Header
Output-Type	calling-e164		
Name	REMOVE-CNPN		
Key	Value	}	Entries
restricted	if USVOIP-A		

Figure 64. 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 19. 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 19. 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.
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 to enable least cost routing.	Cannot be set.
date	Route calls based on the current date to enable least cost routing.	Cannot be set.
day-of-week	Route calls based on the current day of week to enable least cost routing.	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.
3. 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 65 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:

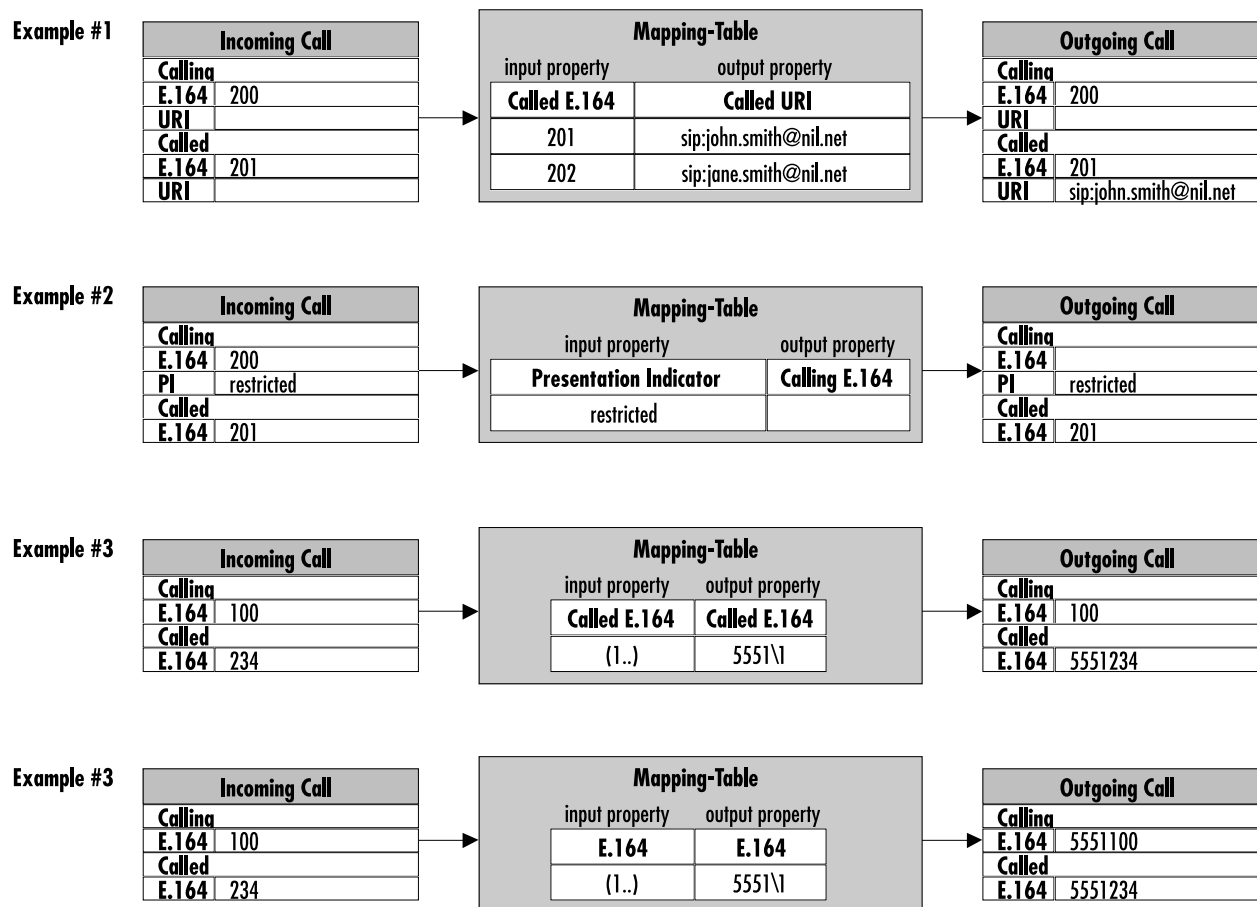


Figure 65. Mapping table examples

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 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)[<i>table-name</i>]#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

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#mapping-table e164 to uri SETURI
SN(rt-tab)[SETURI]#map 100 to sip:john.smith@nil.net
SN(rt-tab)[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:

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#mapping-table calling-e164 to calling-e164 MAP-PSTN
SN(rt-tab)[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

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#mapping-table called-e164 to called-e164 ADD-PFX
SN(rt-tab)[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

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#mapping-table called-e164 to called-e164 REM-PFX
SN(rt-tab)[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) È \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

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#mapping-table called-e164 to called-e164 TRUNC
SN(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) È \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

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#mapping-table calling-pi to calling-e164 REM-CNPN
SN(rt-tab)[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.

Example: Mapping table to replace the calling party number with the called party number

If you route a call to an FXS interface, the Bellcore standard only allows to signal the calling party number to the connected analog terminal instead of the called party number. Thus you cannot communicate e.g. which extension is being called at a destination PBX. This command allows sending the called party number as calling party number property. The called party number still remains the same.

Input:calling-e164 = 0998882525; called-e164 = 0778881111

Output:calling-e164 = 0778881111; called-e164 = 0778881111

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#mapping-table called-e164 to calling-e164 COPY-PN
SN(rt-tab)[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) È \1 = 0778881111

The output (calling party number) is built from the first bracket \1.

The calling party number is set to 0778881111.

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

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#mapping-table called-type-of-number to called-type-of-number
SET-INT
SN(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)

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#mapping-table called-e164 to called-e164 TRANS
SN(rt-tab)[TRANS]#map 550 to 250
SN(rt-tab)[TRANS]#map 551 to 251
SN(rt-tab)[TRANS]#map 552 to 252
SN(rt-tab)[TRANS]#map 553 to 253
SN(rt-tab)[TRANS]#map 554 to 254
SN(rt-tab)[TRANS]#map 555 to 255
SN(rt-tab)[TRANS]#map 556 to 256
SN(rt-tab)[TRANS]#map 557 to 257
SN(rt-tab)[TRANS]#map 558 to 258
SN(rt-tab)[TRANS]#map 559 to 259
```

Note The translation table above can be reduced using regular expressions.

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#mapping-table called-e164 to called-e164 TRANS
SN(rt-tab)[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 on 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:

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#mapping-table MY-TABLE
SN(map-tab)[MY-TABLE]#no map 10
SN(map-tab)[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 table-name</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

```
SN(cfg)#context cs
SN(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 function-name</code>	Create a complex function <i>function-name</i> .
2	<code>node(func)[function-name]#execute function</code> or <code>node(func)[function-name]#execute index function</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

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#complex function CARRIER-TO-LOCAL
SN(func)[CARRIER~]#execute 1 TRUNCATE-3-CNPN
SN(func)[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 on 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 function-name</code>	Enter the complex function from which you want to remove an entry.
2	<code>node(func)[table-name]#no execute index</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:

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#complex-function MY-FUNC
SN(func)[MY-FUNC]#no execute 1
SN(func)[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 <i>function-name</i></code>	Delete the complex function <i>function-name</i> .

Example: Remove an entire complex function

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#no complex-function MY-FUNC
```

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, 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 66 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.

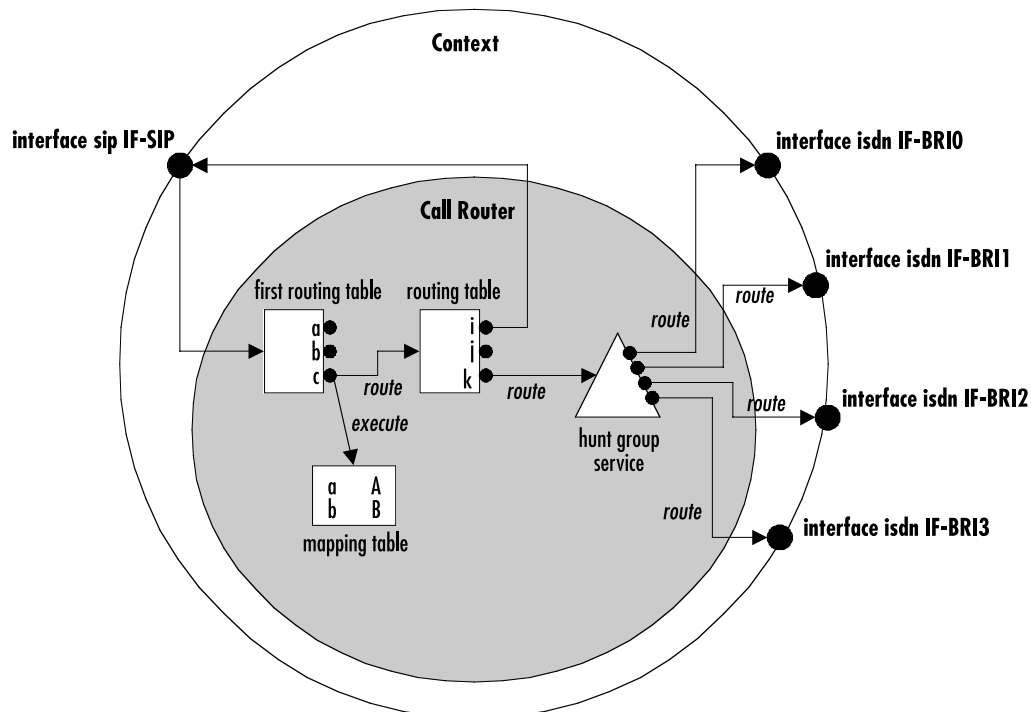


Figure 66. Hunt group service

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 Unlike previous versions of SmartWare, now 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 H.323 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]#timeout 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 20. 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 20. 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 20. Hunt group drop causes (Continued)

Class	Cause	Default Behavior of the Hunt Group Service	Description
Normal Event (Cont.)	response-to-status-enquiry	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 20. 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 20. 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 suspend 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.
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.

Table 20. Hunt group drop causes (Continued)

Class	Cause	Default Behavior of the Hunt Group Service	Description
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.
Inter-working	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 66](#) on page 397.

```

SN(cfg)#context cs
SN(ctx-cs)[switch]#service hunt-group HUNT-BRI
SN(func)[HUNT-BRI]#cyclic
SN(func)[HUNT-BRI]#timeout 6
SN(func)[HUNT-BRI]#route dest-interface IF-BRI0
SN(func)[HUNT-BRI]#route dest-interface IF-BRI1
SN(func)[HUNT-BRI]#route dest-interface IF-BRI2
SN(func)[HUNT-BRI]#route dest-interface IF-BRI3

```

Creating a distribution group service

A distribution group service distributes a call to multiple destination interfaces. Figure 67 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 FXS interfaces. The distribution now lets ring all the four phones at the same time.

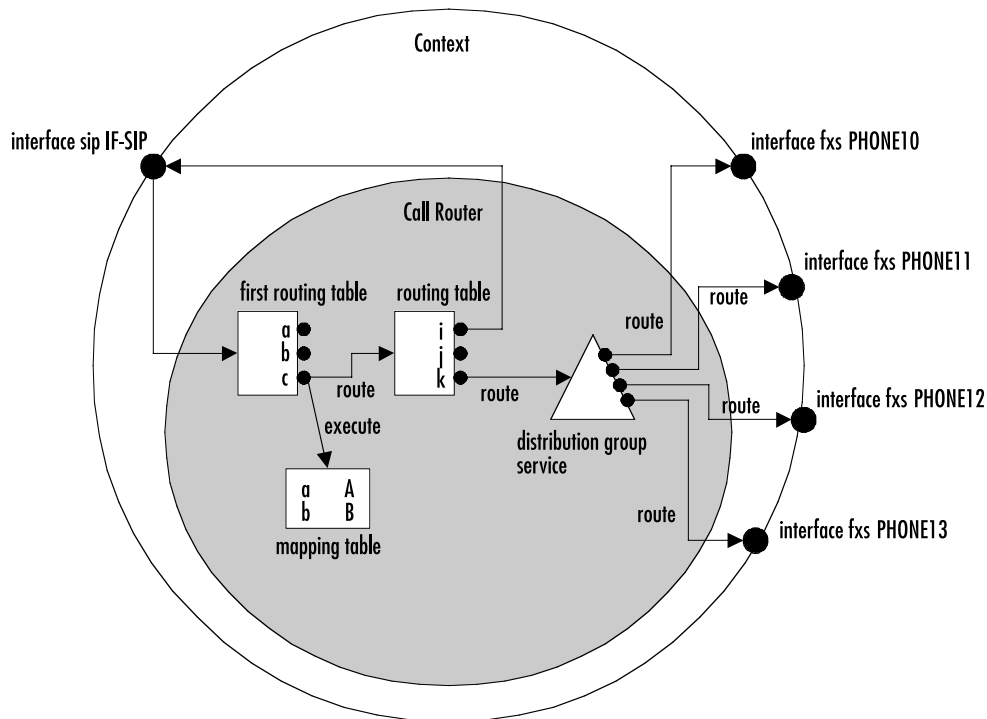


Figure 67. Distribution group service

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 FXS 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.

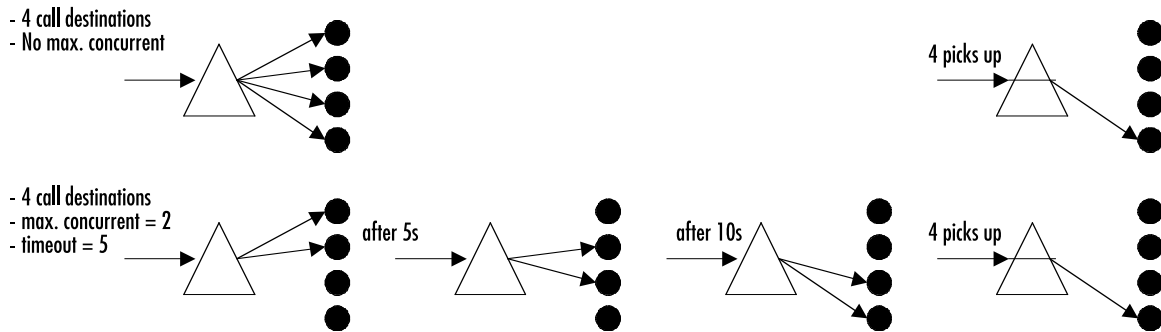


Figure 68. Distribution group service examples

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]#timeout 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.

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

```
SN(cfg)#context cs
SN(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. SmartWare offers a number of possibilities to monitor and debug the CS context and call router configurations. For more information refer to chapter 42, “VoIP debugging” on page 497.

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 Unlike previous versions of SmartWare 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 call-router detail level</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

```
SN(cfg)#context cs
SN(cts-cs)[switch]#routing-table called-e164 TEST
SN(rt-tab)[TEST]#route 1 dest-interface IF1
SN(rt-tab)[TEST]#route 1[0-4] dest-interface IF2
SN(rt-tab)[TEST]#route 11 dest-interface IF3
SN(rt-tab)[TEST]#route 111T dest-interface IF4
```

```

SN(rt-tab)[TEST]#route default dest-interface IF5
SN(rt-tab)[TEST]#exit
SN(ctx-cs)[switch]#no shutdown
SN(ctx-cs)[switch]#debug call-router detail 5
SN(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.

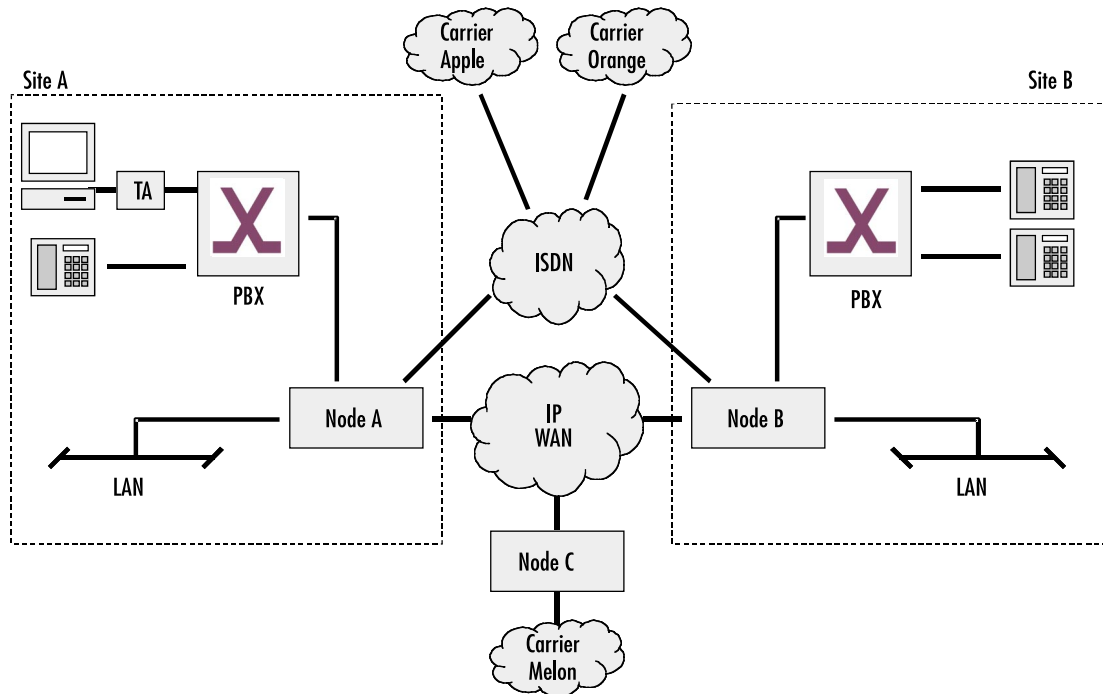


Figure 69. Call routing example network

Note The SmartNodes 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 SmartNode (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.
- H.323 interface *IF-NODE-B*, needed for requirement 2.
- SIP interface *IF-NODE-C*, needed for requirement 3.

Figure 70 shows the corresponding CS Context and call router elements in node A:

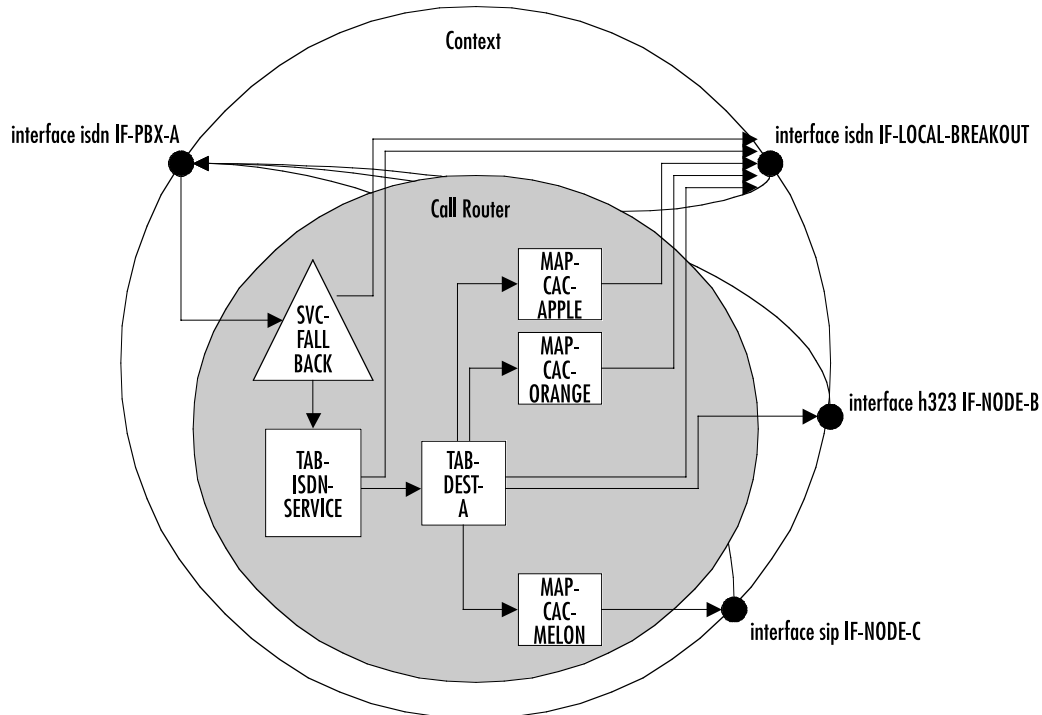


Figure 70. CS context and call router elements

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.

```

SN(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.

```

SN(ctx-cs)[switch]#interface isdn IF-PBX-A
SN(if-pstn)[IF-PBX-A]#route dest-service SVC-FALLBACK
SN(if-pstn)[IF-PBX-A]#exit
SN(ctx-cs)[switch]#interface isdn IF-LOCAL-BREAKOUT
SN(if-isdn)[IF-LOCA~]#route dest-interface IF-PBX-A

```



```
SN(if-isdn)[IF-LOCA~]#exit
SN(ctx-cs)[switch]#interface h323 IF-NODE-B
SN(if-h323)[IF-NODE-B]#route dest-interface IF-PBX-A
SN(if-h323)[IF-NODE-B]#exit
SN(ctx-cs)[switch]#interface sip IF-NODE-C
SN(IF-NODE-C)[IF-NODE-C]#route dest-interface IF-PBX-A
SN(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.

```
SN(cfg)#debug call-router
SN(cfg)#context cs
SN(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'
SN(ctx-cs)[switch]#
```


Chapter 34 **Tone configuration**

Chapter contents

Introduction	418
Tone-set profiles	418
MGCP-Events	419
Tone configuration task list	419
Configuring call-progress-tone profiles	420
Configure tone-set profiles	421
Enable tone-set profile	421
Show call-progress-tone and tone-set profiles	422

Introduction

This chapter gives an overview of SmartWare 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*. Call-progress-tones can be defined for different uses, for tone-set profiles, and for MGCP events.

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 SmartNode. 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 SmartNodes 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 (SmartNodes) 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 SmartNode 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 a supplementary service activated (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 the 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 71 illustrates the relationship between call-progress-tone profiles, tone-set profiles and PSTN interfaces.

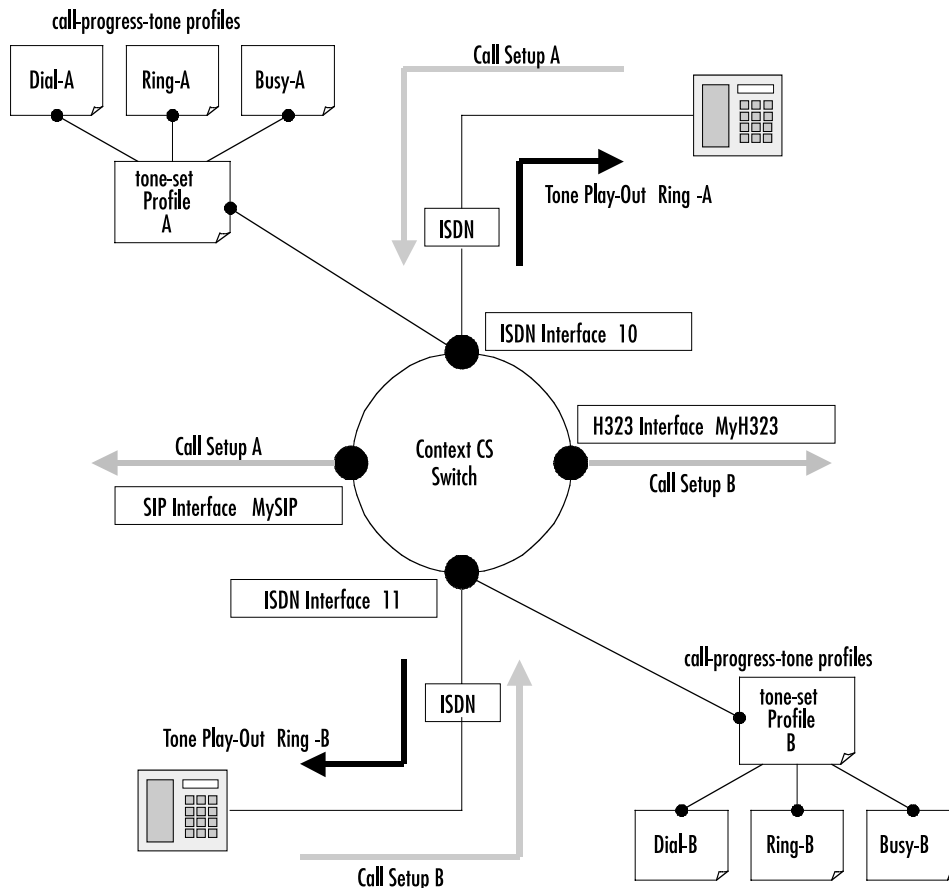


Figure 71. Assign tone-sets to a PSTN interfaces

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.

MGCP-Events

MGCP uses events to request tone playback on the gateway. There are several dozens of events. The MGCP gateway provides a mapping of the MGCP events to configured call-progress-tones.

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. With these parameters all country specific tones can be defined. Tone configuration knows only one command that have 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	node(cfg)#profile call-progress-tone <i>name</i>	Creates a call-progress-tone profile with name <i>name</i> and enters call-progress-tone configuration mode.
2	node(pf-callp)[name]#play <i>duration</i> <i>frequency1 level1 [frequency2 level2]</i>	Defines a tone with duration <i>duration</i> , 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	node(pf-callp)[name]#no play <i>duration</i>	Defines a pause of <i>duration</i> milliseconds
4	node(pf-callp)[name]#...	Repeat step 2 and/or step 3 to define a tone sequence. Always when you enter a play or no play command, it is appended to the already existing tone.
5	node(pf-callp)[name]#flush-play-list	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.

```
SN(cfg)#profile call-progress-tone belgianSpec
SN(pf-callp)[belgian~]#play 330 950 -4
SN(pf-callp)[belgian~]#play 330 1400 -4
SN(pf-callp)[belgian~]#play 330 1800 -4
SN(pf-callp)[belgian~]#no play 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.

Note The play command is new since SmartWare Release 2.10. All your configuration made with previous firmware releases, be it in config files on TFTP servers or stored in the startup-config of the SmartNode, are automatically

converted into the new format. In the running-config you always see the new format, and also new tones have to be configured in the new format. The new tone configuration format is not backwards compatible.

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	node(cfg)#profile tone-set <i>name</i>	Creates tone-set <i>name</i> and enters tone-set profile configuration mode.
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.

Example: Configuring a tone-set

The following example shows how to configure a tone-set profile for UK.

```
SN(cfg)#profile tone-set UK
SN(pf-tones)[UK]#map call_progress_tone dialtone dialUK
SN(pf-tones)[UK]#map call_progress_tone alertingtone ringUK
SN(pf-tones)[UK]#map call_progress_tone busytone busyUK
```

Enable tone-set profile

A call on the SmartNode 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 (H.323, SIP, MGCP). Dependent on the configuration several combinations of signaling protocol endpoints are possible (ISDN–ISDN, H.323-ISDN, FXS-SIP etc.). The SmartNode will always generate the tones locally and play it on the PSTN line as long as the other endpoint doesn't notify availability of inband information and the PSTN endpoint is NOT of type ISDN-USER or FXO. If availability of inband 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 inband information. For enabling a user defined tone-set profile 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 <i>if-type if-name</i>	Enter interface configuration mode.
2	node(if-type)[if-name]#use profile tone-set <i>name</i>	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.

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface isdn bri0
SN(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	node#show profile call-progress-tone [<i>name</i>]	Display all call-progress-tone profiles or a specific with name <i>name</i>

Example: Show call-progress-tone profile

The following example shows how to display the call-progress-tone profiles.

```
SN#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.

```
SN#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:                     defaultConf-tone
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:

```
SN(cfg)#profile call-progress-tone dial-UK
SN(pf-callp)[dial-UK]#play 5000 350 0 440 0

SN(pf-callp)[dial-UK]#profile call-progress-tone alerting-UK
SN(pf-callp)[alertin~]#play 400 400 0 450 0
SN(pf-callp)[alertin~]#no play 200
SN(pf-callp)[alertin~]#play 400 400 0 450 0
SN(pf-callp)[alertin~]#no play 2000

SN(pf-callp)[alertin~]#profile call-progress-tone busy-UK
SN(pf-callp)[busy-UK]#play 400 400 0
SN(pf-callp)[busy-UK]#no play 400
SN(pf-callp)[busy-UK]#exit
```

Create the tone-set profile:

```
SN(cfg)#profile tone-set UK
```

```
SN(pf-tones)[UK]#map call_progress_tone dialtone dial-UK
SN(pf-tones)[UK]#map call_progress_tone alertingtone alerting-UK
SN(pf-tones)[UK]#map call_progress_tone busytone busy-UK
SN(pf-tones)[UK]#exit
```

Assign the tone-set to the isdn interface bri0

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface isdn bri0
SN(if-isdn)[bri0]#use profile tone-set UK
```

Chapter 35 **ISDN port configuration**

Chapter contents

Introduction	426
ISDN reference points	426
Possible SmartNode port configurations	427
ISDN UNI Signaling	427
SmartNode 1000 Series	429
IC-4BRV Interface Card	429
ISDN Configuration Concept	429
ISDN Layering	429
Configuration example	430
Description	430
ISDN port configuration task list	431
Shutdown and enable ISDN ports	431
Configure BRI port parameters (Layer 1)	432
Configure PRI Port Parameters (Layer 1)	433
Configure ISDN layer 2 parameters (Q921)	434
Configure ISDN layer 3 parameters (Q931)	435
Show ISDN port status	437
Examples	439

Introduction

This chapter provides an overview of SmartNode ISDN ports and describes the tasks involved in configuring ISDN ports in SmartWare.

ISDN ports are the physical ISDN connections on the SmartNode 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) physical layer. PRI ports are also called S2m ports. The related PSTN access service is also called Primary Rate Access (PRA).

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 72](#) illustrates these reference points. The understanding of these reference points and where they are located is necessary for the configuration of the SmartNode ISDN ports.

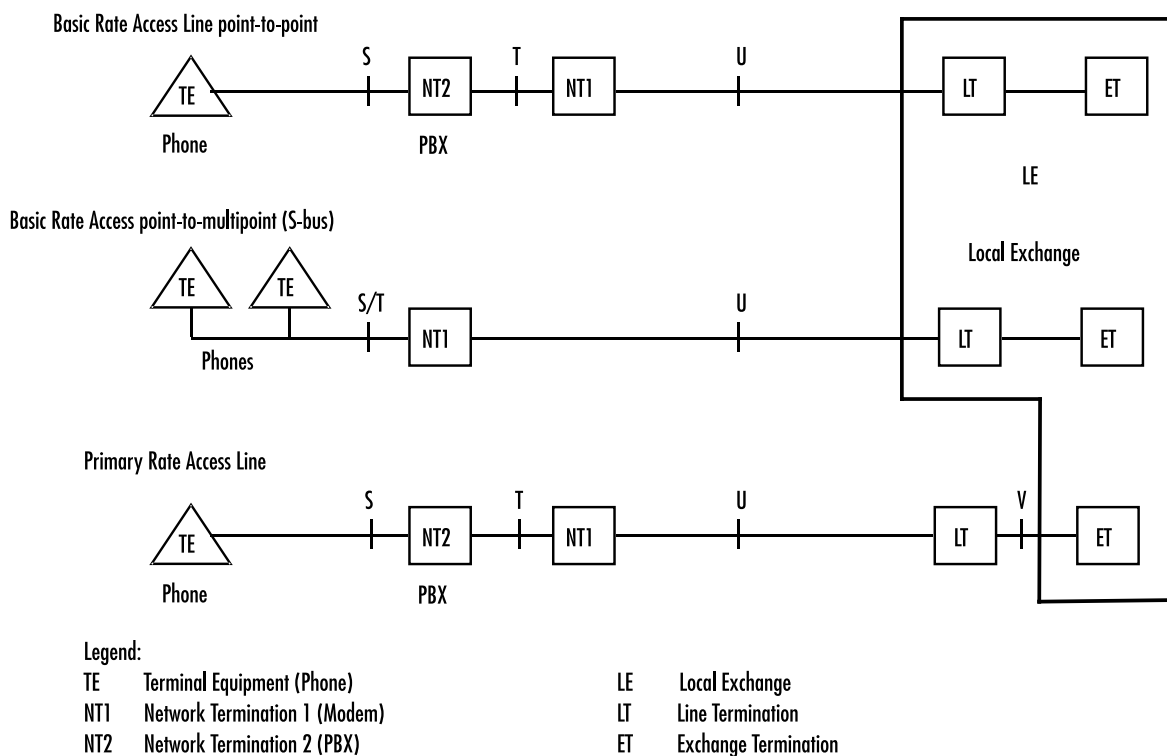


Figure 72. ISDN reference points

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 SmartNode port configurations

The SmartNode ISDN ports can be configured for connection to S, T, S/T, and V interfaces. Refer to [figure 74](#), 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. SmartWare ISDN ports can be configured to work as user (USR) or network (NET) interfaces.

A SmartNode 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 SmartNode ISDN ports are configured to operate the opposite side of the connected equipment as illustrated in [figure 74](#).

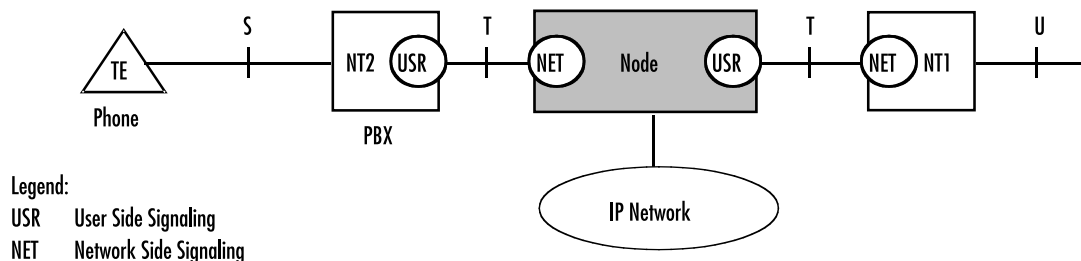


Figure 73. ISDN signaling side

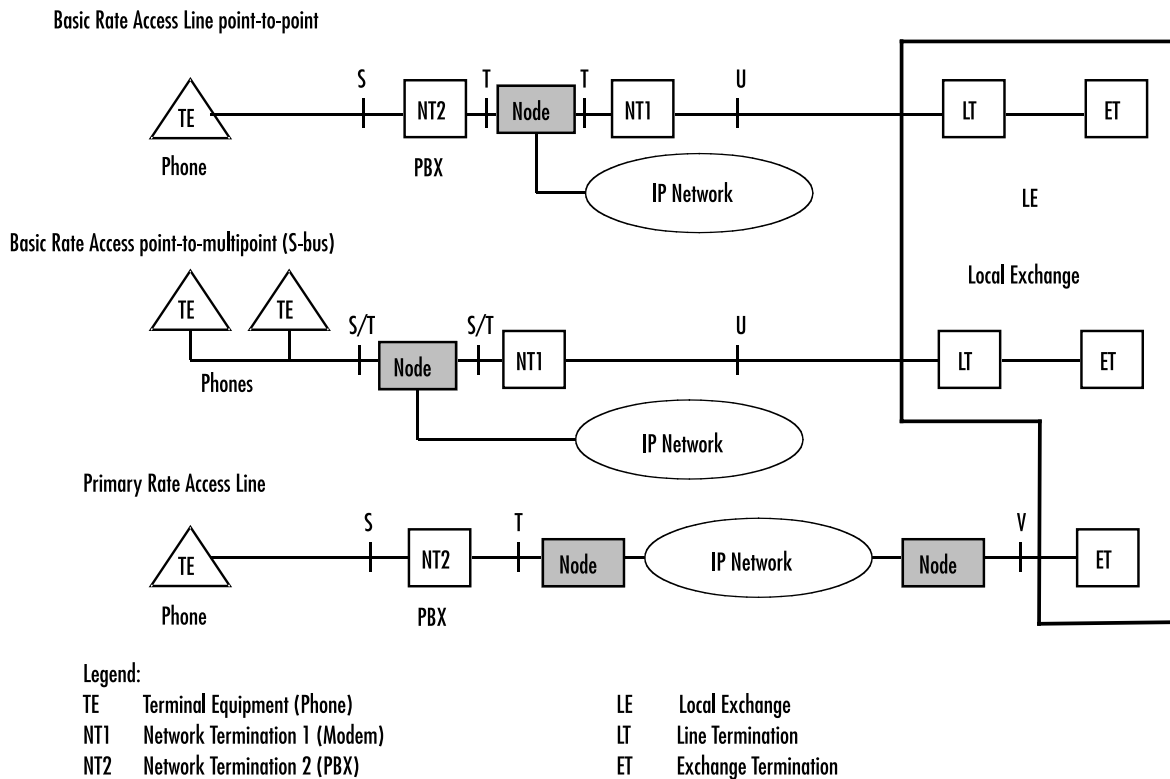


Figure 74. Integration of ISDN access lines



IMPORTANT

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.



IMPORTANT

Reference clock source and synchronization—The SmartNode 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 26, “[CS context overview](#)” on page 293 on how to configure the reference clock



Connector pin-out and short circuits—Some of the SmartNode 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 misconfigurations.

SmartNode 1000 Series

The following table shows which clock mode configurations are allowed for the SmartNode 1000 series and which port is used as the clock source.

Table 21. Clock-modes and clock-sources for the SmartNode 1000 series

Port 0 Clock Mode	Port 1 Clock Mode	Clock Source
Slave (User)	Slave (User)	Port 1 (Sn1400 only)
Master (Net)	Master (Net)	Clock generated internally (Sn1400 only)
Slave (User)	Master (Net)	Port 0. This is the only configuration that is supported on the SmartNode 1200. Clock is taken from port 0 if available. Otherwise it is generated internally.
Master (Net)	Slave (User)	Not supported

Note The SmartNode 1400 needs to be rebooted twice after changing the clock mode.

IC-4BRV Interface Card

With the Interface Card IC-4BRV Version 1 and 2, Ports 0 and 1 can only be Master (Net). Ports 2 and 3 can be either Master (Net) or Slave (User). On IC-4BRV Version 3 all four ports can be either Master (Net) or Slave (User).

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.

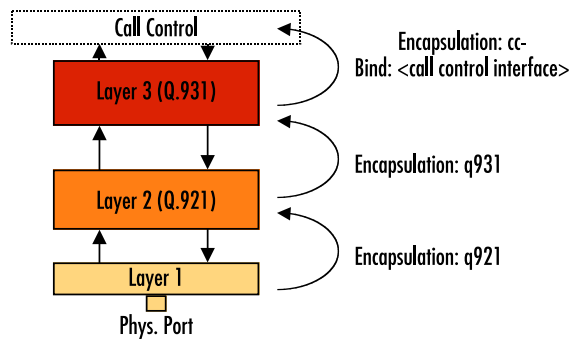


Figure 75. ISDN layering model

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.

Configuration example

Let's have a look at the configuration of a BRI port, running the network side of the DSS1 protocol. To this ISDN port a phone can be attached.

```

1. port bri 0 1
2.   clock auto
3.   encapsulation q921

4. q921
5.   protocol pmp
6.   uni-side auto
7.   encapsulation q931

8. q931
9.   protocol dss1
10.  uni-side net
11.  encapsulation cc-isdn
12.  bind interface isdn01 switch

13. port bri 0 1
14.   no shutdown

```

Description

1. Enter configuration mode for port bri 0/1.
2. Clock mode is automatically determined by layer 2 UNI-side settings.
3. Use 'q921' as layer 2 protocol.

4. Enter Q921 mode.
5. BRI uses point-to-multipoint topology.
6. Layer 2 user/network setting is automatically determined by layer 3 UNI-side settings.
7. Use 'q931' as layer 3 protocol.
8. Enter Q931 mode.
9. Layer 3 uses DSS1 signalling protocol.
10. Running network side part of signalling protocol.
11. Port 0/1 shall be bound to the call control.
12. The call-control interface to use is 'isdn01'.
13. Enter configuration mode for port bri 0/1.
14. Enable port.

ISDN port configuration task list

Configuring ISDN ports typically consists of the following tasks:

- Shutdown and enable ISDN ports
- Configure ISDN layer 1 parameters for BRI ports
- Configure ISDN layer 1 parameters for PRI ports
- Configure ISDN layer 2 parameters
- Configure ISDN layer 3 parameters

Shutdown and enable ISDN ports

By default all ISDN ports are initially shutdown.

Procedure: To disable and activate an ISDN port

Mode: Port {bri|e1|t1}

Step	Command	Purpose
1	<i>node(cfg)#port bri slot port</i>	Enter port configuration. Port type can either be bri, e1 or t1 depending on the installed hardware.
2	<i>node(prt-bri)[slot/port]#shutdown</i>	Shutdown the port. All active calls are dropped!
3	<i>node(prt-bri) [slot/port]#no shutdown</i>	Activate the port.

Example: Shutting down and enabling an ISDN port

The following example shows how to enter the configuration mode for BRI port 0/0, disable and activate the port.

```
172.16.40.71>enable
172.16.40.71#configure
172.16.40.71(cfg)#port bri 0 0
172.16.40.71(prt-bri)[0/0]#shutdown
```

If required change the configuration here.

```
172.16.40.71(prt-bri)[0/0]#no shutdown
```

Configure BRI port parameters (Layer 1)

For a BRI port the only layer 1 setting that can be specified is the clock mode. To work with the port a layer 2 protocol must be bound to it. This is done by specifying an encapsulation.

Procedure: To configure BRI port parameters.

Mode: Port bri

Step	Command	Purpose
1	<code>node(prt-bri)[slot/port]#clock auto</code> or <code>node(prt-bri)[slot/port]#clock master</code> or <code>node(prt-bri)[slot/port]#clock slave</code>	Specify the clock mode (default: auto) The BRI port can either generate the clocking for the line, or accept the clock from the line. The options 'master' or 'slave' determine the clocking method: Master: Generates clock Slave: Accepts clock If the clock mode is not specified or set to auto the setting is derived from the next higher layer. In case of Q921 this is the UNI side setting. NET: clock master UNI: clock slave It is not possible to enable the port unless the clock mode is known.
2	<code>node(prt-bri)[slot/port]#encapsulation q921</code>	Specify the layer 2 protocol to use on this port (default: q921). Currently only Q921 is supported. The no form of the command removes an existing encapsulation.
3	<code>node(prt-bri)[slot/port]#q921</code>	Enter Q921 configuration mode.

Configure PRI Port Parameters (Layer 1)

For the two PRI port types (E1, T1) clocking mode and linecode can be selected. To work with the port a layer 2 protocol must be bound to it. This is done by specifying an encapsulation.

Procedure: To configure PRI port parameters

Mode: Port {e1|t1}

Step	Command	Purpose
1	node(prt-e1)[slot/port]#clock auto or node(prt-e1)[slot/port]#clock master or node(prt-e1)[slot/port]#clock slave	Specify the clock mode (Default: auto) The PRI port can either generate the clocking for the line, or accept the clock from the line. The options 'master' or 'slave' determine the clocking method: Master: Generates clock Slave: Accepts clock If the clock mode is not specified or set to auto the setting is derived from the next higher layer. In case of Q921 this is the UNI side setting. NET: clock master UNI: clock slave It is not possible to enable the port unless the clock mode is known.
2	node(prt-e1)[slot/port]#linecode ami or node(prt-e1)[slot/port]#linecode b8zs or node(prt-e1)[slot/port]#linecode hdb3	Specify the linecode to use (Default: hdb3 for E1, b8zs for T1). Make sure that the linecode matches with the remote system.
3	node(prt-e1)[slot/port]#encapsulation q921	Specify the layer 2 protocol to use on this port (Default: q921). Currently only Q921 is supported. The no form of the command removes an existing encapsulation.
4	node(prt-e1)[slot/port]#q921	Enter Q921 configuration mode.

Configure ISDN layer 2 parameters (Q921)

ISDN Layer 2 (Q921) settings apply to both BRI and PRI ports. They are defined in the q921 mode. To use Q921, the layer 1 encapsulation must be set to q921.

Procedure: To configure ISDN layer 2 (Q921) settings

Mode

q921

Step	Command	Purpose
1	node(q921)[slot/port]#protocol pp or node(q921)[slot/port]#protocol pmp	Specify ISDN layer 2 operating mode (Default: BRI: pmp, PRI: pp). The ISDN layer 2 of 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. PRI ports always run in point-to-point (pp) mode.
2	node(q921)[slot/port]#uni-side auto or node(q921)[slot/port]#uni-side net or node(q921)[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)[slot/port]#[no] permanent-layer2	Enables the ISDN Layer 2 permanently (Default: disabled). By default, the ISDN Layer 2 is not enabled permanently, i.e. the first call enables it. (Currently only available with IC-T1V)
4	node(q921)[slot/port]#encapsulation q931	Specify the ISDN layer 3 protocol (Default: q931) Currently only Q931 is supported. The no form of the command removes an existing encapsulation.
5	node(q921)[slot/port]#q931	Enter Q931 configuration mode.

Configure ISDN layer 3 parameters (Q931)

ISDN Layer 3 (Q931) settings apply to both BRI and PRI ports. They are defined in the q931 mode. To use Q931, the layer 2 encapsulation must be set to q931.

Procedure: To configure ISDN layer 3 (Q931) settings

Mode: q931

Step	Command	Purpose
1	<code>node(q931)[slot/port]#protocol dss1</code> or <code>node(q931)[slot/port]#protocol pss1</code> or <code>node(q931)[slot/port]#protocol ni2</code>	Specify the ISDN layer 3 protocol (Default: BRI: dss1, E1: dss1, T1: ni2) The ISDN layer 3 is the network signaling protocol. SmartWare ISDN ports support Euro-ISDN (E-DSS1), Q.SIG (PSS1) and National ISDN (NI2) signaling. The layer 3 signaling must correspond to the connected ISDN equipment or network.
2	<code>node(q931)[slot/port]#signalling-rule etsi</code> or <code>node(q931)[slot/port]#signalling-rule pss1old</code> or <code>node(q931)[slot/port]#no signalling-rule</code>	Specify channel numbering (Default: etsi) Some older Q-SIG variants make use of a channel numbering scheme that differs from the standard ETSI method. In most cases the ETSI numbering applies. Unless the connected ISDN devices and configured protocols require a different scheme, make sure the numbering is set to ETSI.
3	<code>node(q931)[slot/port]#uni-side net</code> or <code>node(q931)[slot/port]#uni-side user</code>	Specify the UNI side of the interface. If not defined on layer2 (q921 mode) this setting also specifies the UNI side setting for layer2. The layer 2 settings also apply to Q.SIG (PSS1) interfaces. Make sure that the device connected to a SmartNode ISDN port is operating the opposite side of the configured uni-side.
4	<code>node(q931)[slot/port]#max-calls</code> <i>number-of-calls</i> or <code>node(q931)[slot/port]#no max-calls</code>	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.

Step	Command	Purpose
5	node(prt-pstn)[slot/port]#channel-range <i>min max</i> or node(prt-pstn)[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>min</i> and <i>max</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 (T1: 23) time-slots are available for use. The no form of the command restores the default settings.
6	node(q931)[slot/port]# bchan-number-order ascending or node(q931)[slot/port]#bchan-number-order ascending-cyclic or node(q931)[slot/port]#bchan-number-order descending or node(q931)[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 'up' and 'down' modes define whether the time slots are filled at the lowest or highest available slot, i.e. 'up' means that always the lowest available slot is used, 'down' uses always the highest available slot.
7	node(q931)[slot/port]#encapsulation cc-isdn or node(q931)[slot/port]#no encapsulation	Specify encapsulation of port. Currently only CC-ISDN is supported. The no form of the command removes an existing encapsulation.
8	node(prt-e1)[slot/port]#bind interface <i>interface</i>	Bind port to an existing call control interface.

Show ISDN port status

Procedure: To display the current status of an ISDN port

Mode: Exec

Step	Command	Purpose
1	node#show port isdn [<i>slot port</i>] [detail < <i>level</i> >]	Show the status of one or more ISDN ports. If the optional arguments slot/port are omitted the status of all ISDN ports is displayed. <i>Level</i> could be 1..5. Level 1 shows less, level 5 shows all available information. Default level is 3.

Example: Showing status of ISDN port 3/1

The following example shows the current status of ISDN port 1 on a 4BRI card in slot 3.

```

172.16.40.71>enable
172.16.40.71#show port isdn 3 1

Proxy Isdn Driver: BRI 0 3
=====

Slot:                               3
Number of Ports:                     4
Hardware Type:                       BRI
Emergency Mode:                      off
Clock Source Port:                   0

Statistics
-----

Leased buffers:                      25
Max. leased buffers:                 37
Next Call Key:                       0

Proxy Port: BRI 0 3 1
-----

Admin State:                         Open
Real State:                          Open
Operating Layer:                     3

Layer 1
  Clock Mode:                        Master
  Link State:                         up

Layer 2
  Permanent Layer 2:                 off
  Protocol:                          PointToMultiPoint
  UniSide:                           Net

Layer 3
  Protocol:                          Dss1

```



```

UniSide:                Net
MinChannel:             0
MaxChannel:             1
MaxCalls:               2
AIS Blue Alarm:         off
Hunt Mode:              Ascending
Signalling Mode:        Etsi

```

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]#encapsulation cc-isdn
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 SmartNode 1400 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]#encapsulation cc-isdn
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]#encapsulation cc-isdn
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: QSIG

Assume the scenario as illustrated in [figure 76](#):

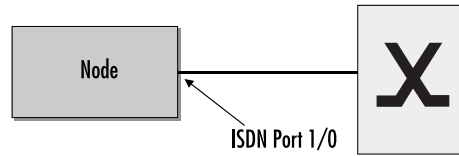


Figure 76. PBX connected to ISDN port 1/0

Configure the ISDN port 1/0 to work as a Q-SIG master port but clock-slave and allow a maximum of eight parallel B-channel connections.

```
172.16.40.71(cfg)#port e1 1 0
172.16.40.71(prt-e1)[1/0]#clock slave
172.16.40.71(prt-e1)[1/0]#q921
172.16.40.71(q921)[1/0]#q931
172.16.40.71(q931)[1/0]#uni-side net
172.16.40.71(q931)[1/0]#protocol pss1
172.16.40.71(q931)[1/0]#signalling-rule etsi
172.16.40.71(q931)[1/0]#max-channels 8
172.16.40.71(q931)[0/0]#exit
172.16.40.71(q921)[0/0]#exit
172.16.40.71(prt-e1)[0/0]#no shutdown
```

Example: PRI

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 e1 1 0
172.16.40.71(prt-e1)[1/0]#q921
172.16.40.71(q921)[1/0]#q931
172.16.40.71(q931)[1/0]#uni-side net
172.16.40.71(q931)[1/0]#max-channels 10
172.16.40.71(q931)[1/0]#channel-range 1 20
172.16.40.71(q931)[1/0]#bchan-number-order descending-cyclic
172.16.40.71(q931)[0/0]#exit
172.16.40.71(q921)[0/0]#exit
172.16.40.71(prt-e1)[0/0]#no shutdown
```

Chapter 36 **FXS port configuration**

Chapter contents

Introduction	442
Shutdown and enable FXS ports.....	442
Bind FXS ports to higher layer applications	443
Configure country-specific FXS port parameters.....	443
Other FXS port parameters.....	444
Choose a low-bit-rate codec for FXS ports.....	445
Example	446

Introduction

This chapter provides an overview of POTS signaling and SmartNode FXS ports and describes the tasks involved in configuring FXS ports in SmartWare.

This chapter includes the following sections:

- Shutdown and enable FXS ports (see [page 442](#))
- Bind FXS ports to higher layer protocols (see [page 444](#))
- Configure country specific FXS port parameters (see [page 443](#))
- Configure other FXS port parameters (see [page 444](#))
- Select a low-bit-rate codec for FXS ports
- Example

FXS stands for *foreign exchange station* and is the exchange side of a POTS (plain old telephone system) line. Even though POTS is seen as 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. POTS signaling.

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 which signals off-hook and on-hook of the handset. Going off-hook establishes a connection between the phone and whatever is on the other side (CO switch, SmartNode, etc.).



IMPORTANT

Connector pinout and short-circuits—The ports of the IC-4FXS use pins 2 and 3 on an RJ-11 jack.



IMPORTANT

Available codecs for IC-4FXS and SN4000 series—Only G.711 and G.723, or G.711 and G.729 are supported on the IC-4FXS and SN4000 series. It is not possible to use G.723 and G.729 simultaneously on these platforms. To choose between G.723 and G.729, see [page 445](#).

Shutdown and enable FXS ports

FXS ports are enabled by default. They also can be configured when they are active. But keep in mind that configuration of an active port temporarily disables the port for a short time (some milliseconds).

Mode: Context CS

Step	Command	Purpose
1	(config)#port fxs <i>slot port</i>	Enter FXS port configuration mode
2	(prt-fxs)[slot/port]#shut-down	Shutdown the port. All active calls are dropped!

Step	Command	Purpose
3	(prt-fxs) [slot/port]#no shut-down	Activate the port.

Bind FXS ports to higher layer applications

An FXS port needs to be associated to an fxs interface in a CS context. The same mechanism of *encapsulation* and *binding* is used as known for e.g. ethernet ports (see “Interfaces, Ports, and Bindings” on page 38).

Procedure: To bind an FXS port to an fxs interface of a CS context

Mode: Port FXS

	Command	Purpose
Step 1	node(config)#port fxs slot port	Enter configuration mode for FXS port
Step 2	node(prt-fxs)[slot/port]#encapsulation cc-fxs	Sets the encapsulation for the port. cc-fxs designs the encapsulation is a context CS interface
Step 3	node(prt-fxs)[slot/port]#bind interface interface	Binds the port to an interface in a CS context

Configure country-specific FXS port parameters

Unlike ISDN, POTS is heavily country specific even though there is a good chance that a phone for one country works reasonably good in another country. Country specific settings are contained in a so-called *fxs profile* which is integrated in the firmware of the SmartNode. It can be updated independently from the firmware by means of tftp download.

As there are more than 190 countries, SmartWare does not support all country parameters, so make sure that the country parameter for your country is available as a profile, before you begin operation.

Procedure: Configure country-specific FXS parameters

Mode: Port FXS

	Command	Purpose
Step 1	node(config)#port fxs slot port	Enter configuration mode for FXS port

	Command	Purpose
Step 2	node(prt-fxs)[slot/port]#use profile fxs profile	<p>Select a profile containing the country specific settings of the port attributes (ring voltage etc.). The available country profiles are listed when entering this command. The names listed are composed as follows:</p> <p>< ISO3166-1-Alpha-2 2 digit country code></p> <p>Examples (currently available profiles):</p> <ul style="list-style-type: none"> • 'ch' for Switzerland • 'us' for the United States of America/Canada • 'gb' for Great Britain • 'etsi' for Europe (ETSI Standard 'EuroPOTS') <p>The default profile (not displayed in the list when entering the command) is etsi.</p>

Other FXS port parameters

This section describes the commands available for the configuration of an FXS port.

Procedure: Configure the FXS port parameters

Mode: Configure

	Command	Purpose
Step 1	node(config)#port fxs <i>slot port</i>	Enter FXS port configuration mode
Step 2 optional	<i>node(prt-fxs) [slot/port]#[no] battery-reversal</i>	Reverses the line polarity at connect/disconnect of the call. This might be required by certain PBX to work correctly. Default: disabled.
Step 3 optional	<i>node(prt-fxs) [slot/port]#end-of-call-signaling</i> { busy-tone { loop-break <duration> } }	Selects the method how SmartNode signals the end of a call to the connected analog terminal, playing a busy-tone or interrupting the loop-current for a certain time (duration in ms). Default: busy-tone.
Step 4 optional	<i>node(prt-fxs)[slot/port]#caller-id format</i> { bell etsi }	Specifies which line protocol is used for caller-id transmission. Use bell for US/Canada, etsi for Europe. Caller-id is enabled/disabled in the higher layer application (interface fxs in context CS). Default: etsi
Step 5 optional	<i>node(prt-fxs)[slot/port]#[no] caller-id attenuation</i> <i>attenuation</i>	Attenuates the modulated caller-id signal (dB). Default: disabled.
Step 6 optional	<i>node(prt-fxs) [slot/port]#flash-hook duration</i> <i>duration</i>	Specifies for what time the connected device goes on-hook for flash-hook signaling (ms). Default: 200ms. In US or canada, try 350ms.
Step 7 optional	<i>node(prt-fxs) [slot/port]#[no] pulse-dialing</i>	Enables the port for use with pulse dialing terminals. Recommended only when a terminal is connected that allows pulse dialing only. Enabling pulse dialing disables the use of all flash-hook features. Default: Disabled.

Choose a low-bit-rate codec for FXS ports

On SmartNode 4000 series and the IC-4FXS card for the 2000 series, it is not possible to use two low-bit-rate codecs at the same time. Thus you must choose to use either G.723 or G.729. G.711 is always supported.

As this limitation only applies to slots with IC-4FXS cards, the command to choose the low-bit-rate codec is in the IC voice mode in the system mode. There are the settings per slot.

Procedure: To choose the low-bit-rate coder

Mode: IC voice in system

	Command	Purpose
Step 1	node(ic-voice)[slot]#low-bit-rate-coder {g729 g723}	Select low-bit-rate coder to be used on the slot <i>slot</i> .

Example

The following example shows how to enter the configuration mode for FXS port 0/0, configure it with typical US settings, and bind it to an interface named *fxs00* in context *CS switch*.

```
172.16.40.71>enable
172.16.40.71#configure
172.16.40.71(cfg)#port fxs 0 0
172.16.40.71(prt-fxs)[0/0]#use profile fxs us
172.16.40.71(prt-fxs)[0/0]#caller-id format bell
172.16.40.71(prt-fxs)[0/0]#flash-hook-duration 350
172.16.40.71(prt-fxs)[0/0]#encapsulation cc-fxs
172.16.40.71(prt-fxs)[0/0]#bind interface fxs00 switch
172.16.40.71(prt-fxs)[0/0]#exit
172.16.40.71(cfg)#system
172.16.40.71(sys)#ic voice 0
172.16.40.71(ic-voice)[0]#low-bit-rate-codec g729
```


Chapter 37 **FXO port configuration**

Chapter contents

Introduction.....	448
Shutdown and enable FXO ports.....	448
Bind FXO ports to higher layer applications.....	449
Configure country specific FXO port parameters.....	449
Other FXO port parameters	450

Introduction

This chapter provides an overview of POTS signaling and SmartNode FXO ports and describes the tasks involved in configuring FXO ports in SmartWare.

This chapter includes the following sections:

- Shutdown and enable FXO ports (see [page 448](#))
- Bind FXO ports to higher layer protocols (see [page 450](#))
- Configure country specific FXO port parameters (see [page 449](#))
- Configure other FXO port parameters (see [page 450](#))
- Example

FXO stands for *foreign exchange office* and is the subscriber side of a POTS (plain old telephone system) line. An FXO port on the SmartNode simulates thus an analog phone set, which must actively go on-hook and off-hook, detect ringing and caller-id, and dial the called-party number using DTMF keypad.



Connector pinout and short-circuits—The ports of the FXO ports use pins 2 and 3 on an RJ-11 jack.



Available codecs for SN4000 series—Only G.711 and G.723, or G.711 and G.729 are supported on the SN4000 series. It is not possible to use G.723 and G.729 simultaneously on these platforms. To choose between G.723 and G.729, see [page 445](#).

Shutdown and enable FXO ports

FXO ports are enabled by default. They also can be configured when they are active. But keep in mind that configuration of an active port temporarily disables the port for a short time (some milliseconds).

Mode: Configure

Step	Command	Purpose
1	(config)#port fxo <i>slot port</i>	Enter FXO port configuration mode
2	(prt-fxo)[slot/port]#shut-down	Shutdown the port. All active calls are dropped!
3	(prt-fxo) [slot/port]#no shut-down	Activate the port.

Bind FXO ports to higher layer applications

An FXO port needs to be associated to an fxo interface in a CS context. The same mechanism of *encapsulation* and *binding* is used as known for e.g. ethernet ports (see section “Interfaces, Ports, and Bindings” on page 38).

Procedure: To bind an FXO port to an fxo interface of a CS context

Mode: Port FXO

	Command	Purpose
Step 1	node(config)#port fxo <i>slot port</i>	Enter configuration mode for FXO port
Step 2	node(prt-fxo)[slot/port]#encapsulation cc-fxo	Sets the encapsulation for the port. cc-fxo designs the encapsulation is a context CS interface
Step 3	node(prt-fxo)[slot/port]#bind interface <i>interface</i>	Binds the port to an interface in a CS context

Configure country specific FXO port parameters

Unlike ISDN, POTS is heavily country specific even though there is a good chance that a setting for one country works reasonably good in another country. Country specific settings are contained in a so-called *fxo profile* which is integrated in the firmware of the SmartNode. It can be updated independently from the firmware by means of tftp download.

As there are over 190 countries SmartWare of course does not support all different country parameters. Thus before operation make sure that the country parameter for your country is available as profile.

Procedure: Configure country specific FXO parameters

Mode: Port FXO

	Command	Purpose
Step 1	node(config)#port fxo <i>slot port</i>	Enter configuration mode for FXO port
Step 2	node(prt-fxo)[slot/port]#use profile fxo <i>profile</i>	<p>Select a profile containing the country specific settings of the port. Available profiles are:</p> <ul style="list-style-type: none"> • ‘etsi’ according to ETSI standard • ‘fcc68_25Hz’ for the United States of America / Canada, according to the FCC86 standard, with 25Hz ringing detection. • ‘fcc68_50Hz’ for the United States of America / Canada, according to the FCC86 standard, with 50Hz ringing detection. • ‘tbr21_25Hz’, according to the TBR 21 standard, with 25Hz ringing detection. • ‘tbr21_50Hz’, according to the TBR 21 standard, with 50Hz ringing detection. <p>The default profile (not displayed in the list when entering the command) is etsi.</p>

Other FXO port parameters

This section describes the commands available for the configuration of an FXO port.

Procedure: Configure the FXO port parameters

Mode: Configure

	Command	Purpose
Step 1	node(config)#port fxo <i>slot port</i>	Enter FXO port configuration mode
Step 2 optional	<i>node(prt-fxo)[slot/port]#caller-id</i> format { bell etsi }	Specifies which line protocol is used for caller-id transmission. Use bell for US / Canada, etsi for Europe. If caller-id is not enabled or wrong configured, detection of caller-id is not possible. Default: etsi
Step 3 optional	<i>node(prt-fxo) [slot/port]#flash-hook</i> <i>duration duration</i>	Specifies for what time the SmartNode should go on-hook to signal flash-hook to the CO. Default of <i>duration</i> is 200ms, for US and Canada try 350ms or higher.

Chapter 38 **H.323 gateway configuration**

Chapter contents

Introduction.....	452
Gateway configuration task list.....	452
Configure datapath related settings	453
Binding the gateway to an IP interface	454
Enable the gateway	454
Configure registration authentication service (RAS) (Optional)	455
Configure H.235 Security (optional)	456
Advanced configuration options (optional)	460
Enabling H.245 Tunneling	460
Enabling the fastconnect procedure	460
Enabling the early H.245 procedure	461
Changing the TCP port for inbound call-signaling connections	461
Setting the response timeout	461
Setting the connect timeout	462
Configuring the terminal type for registration with the gatekeeper	462
Troubleshooting	463

Introduction

This chapter provides an overview of the H.323 gateway and describes the tasks involved in its configuration.

When communication is required between different networks a gateway is always needed between them. A gateway provides:

- Data format translation, e.g. audio and video CODEC translation
- Control signaling translation, e.g. call setup and termination functionality on both sides of a network.

In the case of SmartWare, a gateway connects two contexts of different types, For example, the CS and the IP context. It handles connections between different technologies or protocols and contains general gateway configuration parameters. In SmartWare one of these gateways is the H.323 gateway. This is an implementation of the ITU-T H.323 Version 4 standard. The H.323 uses H.323 CS interfaces, which are explained in detail in the chapter about H.323 Interface Configuration. The H.323 interfaces in the CS context must be explicitly bound to a H.323 gateway instance. The H.323 gateway itself must be bound explicitly to an IP interface in the IP context. [Figure 77](#) illustrates the function of the H.323 gateway. SmartWare currently supports only one instance of the H.323 gateway. The default name of the H.323 gateway is h323.

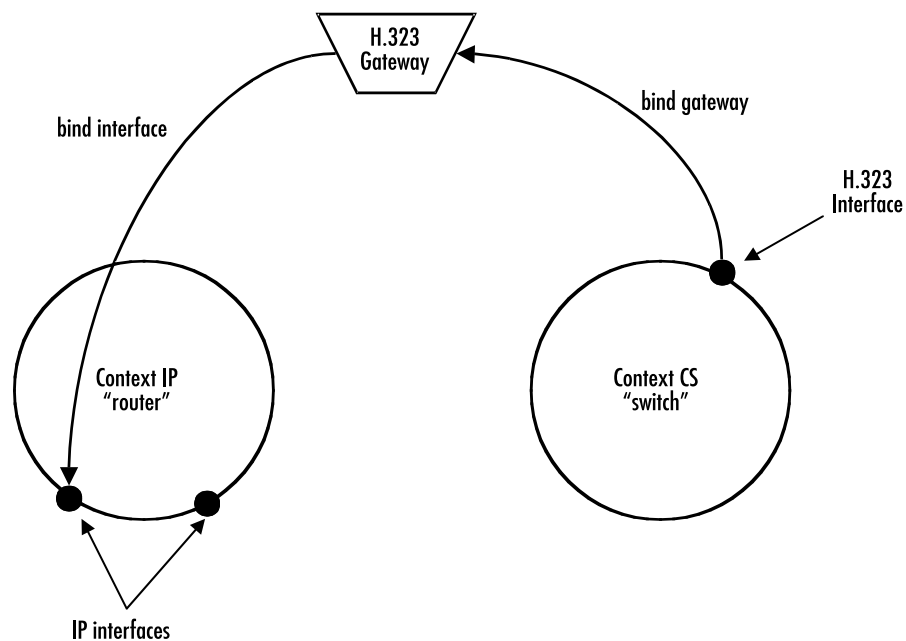


Figure 77. Gateway between IP and CS contexts

Gateway configuration task list

This chapter describes the configuration of the H.323 gateway. Some parameters can be configured in the gateway configuration mode and may be overwritten in another configuration mode, For example, in the H.323 CS interface. For example, the default VoIP profile to be used with the gateway is configured in the gateway configuration mode. However it can be overwritten by another VoIP profile specified in the H.323 CS inter-

face. All possible configurations, which are involved in a specific configuration topic are described in the respective configuration task.

- Configure datapath related settings
- Binding the gateway to an IP interface
- Enable the Gateway
- Configure Registration Authentication Service (RAS) (Optional)
- Configure H.235 Security (Optional)
- Advanced configuration options (Optional)

Configure datapath related settings

To retrieve configuration data about allowed codecs and other datapath related features such as T.38 and fax or modem bypass, the H.323 gateway uses a VoIP profile. You can find detailed information about how to create a VoIP profile in the chapter about VoIP profile configuration of this configuration guide. The following procedure describes how to use an existing VoIP profile with the gateway. If this configuration step is omitted, the default VoIP profile named 'default' will be used automatically.

The most important information configured in the VoIP profile is the list of allowed audio codecs.

The VoIP profile defined in the gateway will be overridden by an optional VoIP profile defined in the CS interface.

In H.323 there are two different procedures for media channel negotiation. The two procedures are normal H.245 and fastconnect. Depending on the procedure used, the negotiation of audio codecs slightly differs.

If the normal H.245 procedure is used, both peers of the H.323 connection send the codecs, which are configured in the VoIP profile used for the call to the remote gateway. The remote gateway then selects one codec of the received list, which it also supports and opens a media channel using that codec. Because each peer selects the codec to be used independent of the other peer, it is possible that not both peers do select the same type of codec. This would result in different types of codecs to be used for the receiving and the transmitting media channel of a gateway. It must be ensured by configuration that both directions use the same type of codec, because different types of codecs are not supported by the gateways. If different codecs would be negotiated the call could not be established. Since this problem does not occur with the fastconnect procedure, it is recommended to use the fastconnect procedure if possible. Also the normal H.245 procedure requires about 10 packets to be exchanged by the peers in order to open the media channels while the fastconnect procedure requires no additional messages to be exchanged.

When using the fastconnect procedure, the gateway which initiates the H.323 call, sends the list of codecs defined in the VoIP profile for the call as an offer to the remote gateway. The gateway, which is the destination, selects the first entry of the received codec list that is also included in the VoIP profile that itself will use for the call. The codec selected by that gateway will then be used by both gateways to open a media channel of that type. This ensures that both media channels of the call always use the same codec.

Note The VoIP profile defined in the gateway will be overridden by an optional VoIP profile defined in the CS interface.

Note If no codecs are configured in the codec list of the VoIP profile, you will be unable to establish calls over the gateway.

Mode: Gateway H.323

Step	Command	Purpose
1	<code>node(gw-h323)[h323]#use profile voip profile-name</code>	Use an existing VoIP profile for this gateway.

Example: Using a VoIP profile

The following example shows how to use the existing VoIP profile myvoip for the gateway.

```
SN>enable
SN#configure
SN(cfg)#gateway h323 h323
SN(gw-h323)[h323]#use profile voip myvoip
```

Binding the gateway to an IP interface

Binding the gateway to one of the available IP interfaces is required to allow the gateway to determine the local IP address it should use. The gateway needs to know the local IP address. For example, when it needs to tell the remote gateway in the call-signaling to which IP address the remote gateway should send the media streams (RTP data). The gateway always uses the IP address of the interface to which it is bound, when a local IP address is required. The following procedure describes how to bind the gateway to a local IP interface.

Mode: Gateway H.323

Step	Command	Purpose
1	<code>node(gw-h323)[h323]#bind interface if-name</code>	Binds the gateway to the IP interface if-name.

Example: Binding the gateway

The following example shows how to bind the gateway to an IP interface.

```
SN>enable
SN#configure
SN(cfg)#gateway h323 h323
SN(gw-h323)[h323]#bind interface eth0
```

Enable the gateway

In order to become active the H.323 gateway must be enabled. The following procedure enables the H.323 gateway:

Mode: Gateway H.323

Step	Command	Purpose
1	<code>node(gw-h323)[h323]#no shutdown</code>	Enable the H.323 gateway

Example: Enabling an H.323 gateway

The following example shows how to enable an already defined H.323 Gateway.

```
SN(cfg)#gateway h323 h323
SN(gw-h323)[h323]#no shutdown
```


Configure registration authentication service (RAS) (Optional)

The H.323 gateway can either work in combination with a gatekeeper, which uses the RAS protocol for communication with the gateways or it can be used for direct calls between gateways without a gatekeeper. If you do not use a gatekeeper, you can skip this section. Otherwise you need to provide some information for the registration with the gatekeeper using the procedures described in this chapter.

The SmartNode can register one or more names, E.164 numbers or gatekeeper prefixes, which can be specified using this procedure too. Furthermore the gatekeeper discovery method must be specified as automatic or manual.

Normally the remote IP address which could be specified in the CS interface is not set if gatekeepers are used, because the gatekeeper supplies the remote destination IP address.

Redundancy is of great importance in communication networks. An H.323 gatekeeper offers a critical service in a network – failure of a single gatekeeper may result in non-availability of the voice service in the entire network. The SmartNode allows configuring up to three different gatekeepers, when using the manual gatekeeper discovery method. These gatekeepers are tried in a round-robin fashion. Once communication with one of the gatekeepers is lost the SmartNode falls back to the next gatekeeper in the list.

Procedure: To enable the registration authentication service (RAS)

Mode: Gateway H.323

Step	Command	Purpose
1	<code>node(gw-h323)[h323]#gatekeeper-discovery auto [gkid]</code> or <code>node(gw-h323)[h323]#gatekeeper-discovery manual ip-address ip-port [gkid]</code>	Specify that gatekeeper discovery has to be done automatically or Specify the gatekeeper for the SmartNode explicitly. You can repeat this command to add multiple gatekeepers.
2	<code>node(gw-h323)[h323]#alias h323-id name</code> or <code>node(gw-h323)[h323]#alias e164 number</code>	Add an H.323_ID or E.164 alias for registration.
3		Repeat step 2 to add more than one alias to the configuration.
4	<code>node(gw-h323)[h323]#supported-prefix prefix</code>	Add a gatekeeper prefix to be registered on the gatekeeper
5		Repeat step 4 to add more than one prefix to the configuration.
6	<code>node(gw-h323)[h323]#ras</code>	Enable the RAS protocol to make use of gatekeepers and initiate gateway registration.

Example: Configuring RAS

The following example shows how to configure the registration authentication service (RAS) for a SmartNode in an H.323 network.

```
SN>enable
SN#configure
SN(cfg)#gateway h323 h323
SN(gw-h323)[h323]#gatekeeper-discovery auto
SN(gw-h323)[h323]#alias h323-id Bernel
SN(gw-h323)[h323]#alias e164 007
SN(gw-h323)[h323]#alias e164 *5
SN(gw-h323)[h323]#alias e164 19421
SN(gw-h323)[h323]#supported-prefix 03198525
SN(gw-h323)[h323]#ras
```

Configure H.235 Security (optional)

H.235 is an ITU-T Recommendation for security and encryption for H-series (H.323 and other H.245-based) multimedia terminals. It describes enhancements within the framework of the H.3xx-Series Recommendations to incorporate security services such as Authentication and Privacy (data encryption).

SmartWare implements H.235 Annex D, which provides H.323 RAS and H.225 message authentication and integrity check thus thwarting any replay and spoofing attacks on H.323 calls. If H.235 is switched on, the following security attacks are thwarted:

- Denial of Service attacks
- Man-in-the-middle attacks
- Replay attacks (replay of recorded messages)
- Spoofing
- Connection hijacking

Among other information such as time stamp, sender and general ID, the H.235 needs a password for crypto token generation. Since this password is intelligible when being configured by means of a Telnet session or displayed in a running configuration, it is possible to configure an encrypted password, which will be decrypted on the SmartNode. For decryption a master password is needed. Configuration of the master password should not be done over insecure links (links subject to wire-tapping). It is recommended to do so in a secure network (local area network) only (before delivery to the customer).

Henceforth, the H.235 password can be reconfigured securely even over insecure links.

To generate an H.235 encrypted password by means of the master password as key, the password encryption tool is used ('getcryptopassword.exe'). The usage of the Windows based command line tool is as follows:

```
getcryptopassword <h235-password> <master-password>
```

The H.235 password must be a random alphanumeric character string of 1 through 12 characters (e.g. 12ygR34230kG). The master password must be a 32 digit hex number (characters 0-9, a-f). To achieve best encryption security, choose a random value (no repeating character sequences). The tool generates the encrypted H.235 password and the hash of the master password. The encrypted H.235 password is then to be used for remote (over insecure link) configuration of the H.235 password. The hash value of the master password can be used to verify proper configuration of all parameters. The command 'show h235security' displays all H.235 settings including a hash value of the master password. If this value is identical to the hash value output by the tool 'getcryptopassword.exe', the configuration of the master password was successful. Note that this last verification step can be done securely even over insecure links (subject to wire-tapping) since the algorithm used for hash value calculation is a mathematical one-way function (virtually impossible to derive the password from the hash value). To enable H.235 security on H.323 perform the steps described below.

Procedure: To enable H.235 security on H.323 gateway

Mode: Gateway H.323

Step	Command	Purpose
1	<code>node(gw-h323)[h323]#h235security master-password master-password</code>	Sets the master password (32 hex digits, 0-9, A-F) with which the H.235 password is decrypted. Note: configure the master password only over secure links (e.g. in LAN environments only or with serial connection), which cannot be wire-tapped.
2	<code>C:\getcryptopassword h235-password master-password</code>	Generates H.235 password by means of the master password with the encryption tool.
3	<code>node(gw-h323)[h323]#h235security password h235-password encrypted</code> or <code>node(gw-h323)[h323]#h235security password h235-password</code>	Sets the password used for crypto token calculation. The password is entered encrypted. The password to be entered is the output of the tool 'getcryptopassword.exe'. Configures the password used for crypto hashed token calculation. The password is entered in clear text (min. 1, max. 12 alphanumeric characters). N.B.: Do not use this command over insecure links (subject to wiretapping). If you enter the password as clear text, you don't need to configure a master-password.
4	<code>node(gw-h323)[h323]#h235security time-window time-window</code>	Sets the time window used for timestamp comparison by H.235. If a received H.323 message with H.235 crypto token has a timestamp outside the time window (relative to the local time) the message is refused.
5	<code>node(gw-h323)[h323]#h235security version {v1 v2}</code>	General there exist two H.235 versions. Use 'v1' if 'v2' does not work. In 'v1' "sender-id" and "general-id" must not be specified.
6	<code>node(gw-h323)[h323]#h235security sender-id sender-id</code>	Sets the ID of the Node. Must be a string containing 2, 4, 6, ... characters.
7	<code>node(gw-h323)[h323]#h235security general-id general-id</code>	Sets the ID of the entity to which the message is sent, e.g. a gatekeeper. Must be a string containing 2, 4, 6, ... characters.
8	<code>node(gw-h323)[h323]#h235-security ras-auth-int-rx</code> or <code>node(gw-h323)[h323]#no h235-security ras-auth-int-rx</code>	Enables or disables H.235 security for received RAS packets.

Step	Command	Purpose
9	<code>node(gw-h323)[h323]#h235-security ras-auth-int-tx</code> <i>or</i> <code>node(gw-h323)[h323]#no h235-security ras-auth-int-tx</code>	Enables or disables H.235 security for transmitted RAS packets
10	<code>node(gw-h323)[h323]#h235-security q931-auth-int</code> <i>or</i> <code>node(gw-h323)[h323]#no h235-security q931-auth-int</code>	Enables or disables H.235 security for call-signaling.
11	<code>node(gw-h323)[h323]#h235security</code>	If all parameters are set, enables H.235 security. Otherwise returns an error message.
12	<code>node(gw-h323)[h323]#show h235security</code>	Shows status of H.235 security module.
13	<code>node(gw-h323)[h323]#debug h235security [detail debug-level]</code>	Enables the H.235 debug monitor to display information for every received and sent H.323 signaling message.

Example: Switch on H.235 security

The following example shows how to use the password encryption tool and how to enable H.235 security. Additionally the H.235 security debug monitor is enabled.

Generate the encrypted H.235 password from myh235pwd :

```
C:\>getcryptopassword myh235pwd 12d3f4e76a83c6dd1067a6d34fe5cb21
```

```
H.235 Password           : myh235pwd
Encrypted H.235 Password: 21dafa5dfc7399e5cef9cc138dabd22f
Master Password         : 12d3f4e76a83c6dd1067a6d34fe5cb21
Hash of Master Password : bc210d2244a1afd2e7da7a54a1a8c179403220c4
```

```
C:\>
```

Configure and enable H.235:

```
172.16.224.102(cfg)#gateway h323 h323
172.16.224.102(gw-h323)[h323]#h235security master-password
12d3f4e76a83c6dd1067a6d34fe5cb21
172.16.224.102(gw-h323)[h323]#h235security password
21dafa5dfc7399e5cef9cc138dabd22f encrypted
172.16.224.102(gw-h323)[h323]#h235security time-window 100
172.16.224.102(gw-h323)[h323]#h235security version v2
172.16.224.102(gw-h323)[h323]#h235security sender-id NODE13
```

```
172.16.224.102(gw-h323)[h323]#h235security general-id GK01
172.16.224.102(gw-h323)[h323]#h235security ras-auth-int-tx
172.16.224.102(gw-h323)[h323]#h235security ras-auth-int-rx
172.16.224.102(gw-h323)[h323]#h235security q931-auth-int
172.16.224.102(gw-h323)[h323]#show h235security
```

H.235 SECURITY SETTINGS

```
H.235 Security           : Disabled
H.235 Module Version    : 2.02.01
H.235 Version           : 2
Sender ID                : NODE13
General ID               : GK01
Time Window              : 100 seconds
Master Password Hash    : bc210d2244a1afd2e7da7a54a1a8c179403220c4
Debug Monitor           : Disabled
```

```
172.16.224.102(gw-h323)[h323]#
172.16.224.102(gw-h323)[h323]#debug h235security detail 3
172.16.224.102(gw-h323)[h323]#h235security
172.16.224.102(gw-h323)[h323]#14:27:35 H235 > Info: H.235 was started successfully
```

Advanced configuration options (optional)

Enabling H.245 Tunneling

If H.245 Tunneling is enabled, H.245 messages use the same TCP connection as the H.323 signaling. H.245 Tunneling is disabled by default. If enabled, it only takes place when both parties agree. When experiencing problems with establishing H.323 calls, disabling H.245-Tunneling may help.

Mode: Gateway H.323

Step	Command	Purpose
1	<code>node(gw-h323)[h323]#h245-tunneling</code>	Enables H.245 tunneling.

Example: Enabling H.245 tunneling

The following example shows how to enable H.245 tunneling on an already defined H.323 Gateway.

```
SN(cfg)#gateway h323 h323
SN(gw-h323)[h323]#h245-tunneling
```

Enabling the fastconnect procedure

If the fastconnect procedure is enabled, no separate H.245 connection is opened and all media channel specific messages are carried within the H.225 call signaling connection. Fastconnect is disabled by default. If enabled, it only takes place when both parties agree. Fastconnect can be enabled using the following procedure:

Mode: Gateway H.323

Step	Command	Purpose
1	<code>node(gw-h323)[h323]#faststart</code>	Enables the fastconnect procedure.

Example: Enabling fastconnect

The following example shows how to enable the fastconnect procedure on an already defined H.323 Gateway.

```
SN(cfg)#gateway h323 h323
SN(gw-h323)[h323]#faststart
```

Enabling the early H.245 procedure

If the early H.245 procedure is enabled, the H.245 connection is opened as soon as possible instead of waiting for the call signaling connect message. Early H.245 is disabled by default. If enabled, it only takes place when both parties agree. The early H.245 procedure can be enabled using the following procedure:

Mode: Gateway H.323

Step	Command	Purpose
1	<code>node(gw-h323)[h323]#early-h245</code>	Enables the early H.245 procedure.

Example: Enabling early H.245

The following example shows how to enable early H.245 on an already defined H.323 Gateway.

```
SN(cfg)#gateway h323 h323
SN(gw-h323)[h323]#early-h245
```

Changing the TCP port for inbound call-signaling connections

The default TCP listening port for inbound call-signaling connections is per default 1720 as defined in the H.323 standard. The following procedure describes how to change the port number.

Mode: Gateway H.323

Step	Command	Purpose
1	<code>node(gw-h323)[h323]#call-signaling-port port</code>	Define the TCP port to use for inbound call-signaling connections

Example: Defining an alternate call-signaling port

The following example shows how to define an alternate call-signaling port on an already defined H.323 Gateway.

```
SN(cfg)#gateway h323 h323
SN(gw-h323)[h323]#call-signaling-port 1721
```

Setting the response timeout

Per default the H.323 gateway waits for 12s from the time it initiated a call towards the IP network until it terminates the call, if no response has been received. This time can be changed using the following procedure:

Mode: Gateway H.323

Step	Command	Purpose
1	<code>node(gw-h323)[h323]# timeout response seconds</code>	Defines the response timeout in seconds.

Example: Defining an alternate response timeout

The following example shows how to define an alternate response timeout of 6 seconds on an already defined H.323 Gateway.

```
SN(cfg)#gateway h323 h323
SN(gw-h323)[h323]#timeout response 6
```

Setting the connect timeout

Per default the H.323 gateway waits for 60s when the call is in the alerting phase for the call to be answered. If the call is not answered within that time, the call is dropped. The value of this timer can be changed using the following procedure:

Mode: Gateway H.323

Step	Command	Purpose
1	<code>node(gw-h323)[h323]# timeout connect sec- onds</code>	Defines the connect timeout in seconds.

Example: Defining an alternate connect timeout

The following example shows how to define an alternate connect timeout of 20 seconds on an already defined H.323 Gateway.

```
SN(cfg)#gateway h323 h323
SN(gw-h323)[h323]#timeout connect 20
```

Configuring the terminal type for registration with the gatekeeper

H.323 gatekeepers may differentiate between two terminal types (terminals and gateways) of the registrant. In some applications it is necessary for the gateway to register as a terminal, while in other applications it is necessary to register as a gateway with the gatekeeper. The default terminal type is gateway. It can be changed using the following procedure: Usually you do not need to change this setting.

Procedure: Configure the registration type of the registration with the gatekeeper

Mode: Gateway H.323

Step	Command	Purpose
1	<code>node(gw-h323)[h323]#terminal-type { termi- nal gateway }</code>	Set the registration type of the gatekeeper registration.

Example: Configuring RAS registration type

The following example shows how to configure the RAS registration type.

```
SN(cfg)#gateway h323 h323
SN(gw-h323)[h323]#terminal-type gateway
```


Troubleshooting

You can display basic status information of the H.323 gateway using the following command:

Mode: Configure

Step	Command	Purpose
1	<code>node(gw-h323)[h323]# show gateway h323 status [detail level]</code>	Displays H.323 gateway status information. The detail <i>level</i> parameter is a number in the range 0 to 5 and indicates how much detail should be displayed.

Example: Display H.323 gateway status information

The following example shows how to display H.323 gateway status information and a sample output of the command.

```
SN(cfg)#show gateway h323 status
      H.323 Gateway:                h323
      State:                        UP
      Stack Handle:                  0xb6a70c
      RAS Engine
      State:                        UNREGISTERED
      Allocated Endpoints:           0
      Allocated RAS Engines:         1
      Allocated Control Channels:    0
      Allocated Outgoing Logical Slowstart Channels: 0
      Allocated Outgoing Logical Faststart Channels:0
      Allocated Incoming Logical Channels:0
```

The H.323 gateway also provides several debugging monitors to observe its dynamic behavior. These monitors allow efficient troubleshooting of H.323 problems. The most often used monitors are listed in the following table.

Command to enable the monitor	Output of the monitor
<code>node(cfg)#debug gateway h323 error</code>	Logs all errors detected within the H.323 gateway
<code>node(cfg)#debug gateway h323 tpktchan</code>	Logs all H.225 call signaling messages sent or received over the IP network.
<code>node(cfg)#debug gateway h323 udpchan</code>	Logs all H.225 RAS messages sent or received over the IP network
<code>node(cfg)#debug gateway h323 datapath</code>	Logs information related to media channels
<code>node(cfg)#debug gateway h323 signaling</code>	Logs call signaling related information

Chapter 39 **SIP gateway configuration**

Chapter contents

Introduction	466
Gateway configuration task list	466
Configure DNS resolver	467
Configure datapath related settings	467
Binding the gateway to an IP interface	468
Enable the Gateway	468
Registering with a registrar (optional)	468
Configure a domain name (optional)	469
Configure a default server (optional)	470
Configure authentication parameters (optional)	471
Enable the session timer (optional)	471
Advanced configuration options (optional)	472
Changing the listening port for inbound call-signaling	472
Define session timer version	472
Define call transfer version	473
Troubleshooting	473

Introduction

This chapter provides an overview of the SIP gateway and describes the tasks involved in its configuration.

When communication is required between different networks a gateway is always needed between them. A gateway provides:

- Data format translation, e.g. audio and video CODEC translation
- Control signaling translation, e.g. call setup and termination functionality on both sides of a network.

In the case of SmartWare, a gateway connects two contexts of different types, For example, the CS and the IP context. It handles connections between different technologies or protocols and contains general gateway configuration parameters. In SmartWare one of these gateways is the SIP gateway. This is an implementation of a Session Initiation Protocol (SIP) User Agent according to RFC 3261. The SIP gateway uses SIP CS interfaces, which are explained in detail in the chapter about SIP Interface Configuration. The SIP interfaces in the CS context must be explicitly bound to a SIP gateway. The SIP gateway itself must be bound explicitly to an IP interface in the IP context. [Figure 78](#) illustrates the function of the SIP gateway. SmartWare supports multiple instances of the SIP gateway.

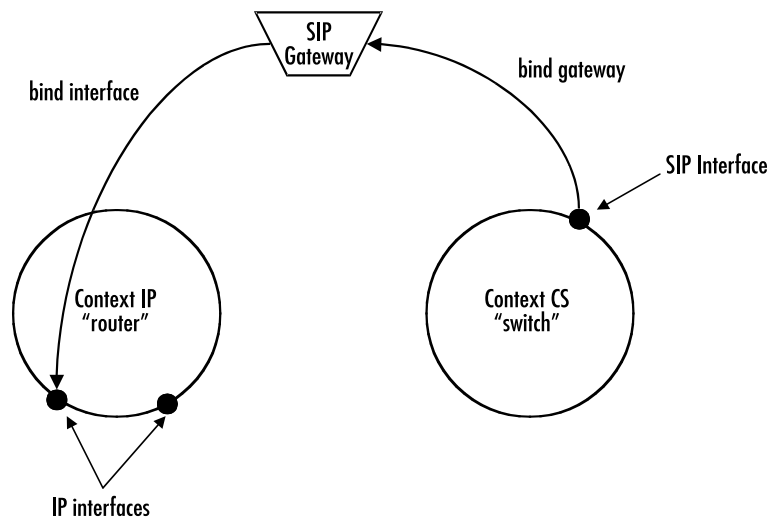


Figure 78. SIP Gateway between IP and CS contexts

Gateway configuration task list

This chapter describes the configuration of the SIP gateway. Some parameters can be configured in the gateway configuration mode and may be overwritten in another configuration mode, For example, in the SIP CS interface. For example, the default VoIP profile to be used with the gateway is configured in the gateway configuration mode. However it can be overwritten by another VoIP profile specified in the SIP CS interface. All possible configurations, which are involved in a specific configuration topic are described in the respective configuration task.

- Configure DNS resolver
- Configure datapath related settings
- Binding the gateway to an IP interface

- Enable the Gateway
- Registering with a registrar (Optional)
- Configure a domain name (Optional)
- Configure a default server (Optional)
- Configure authentication parameters (Optional)
- Enable the session timer (Optional)
- Advanced configuration options (Optional)

Configure DNS resolver

The SIP gateway requires the DNS resolver to be enabled. You can find additional information about how to configure the DNS resolver in the DNS configuration chapter of this configuration guide.

Configure datapath related settings

To retrieve configuration data about allowed codecs and other datapath related features such as T.38 and fax or modem bypass, the SIP gateway uses a VoIP profile. You can find detailed information about how to create a VoIP profile in the chapter about VoIP profile configuration of this configuration guide. The following procedure describes how to use an existing VoIP profile with the gateway. If this configuration step is omitted, the default VoIP profile named 'default' will be used automatically.

The most important information configured in the VoIP profile is the list of allowed audio codecs.

In SIP the initiator of a call always sends an offer containing all the codecs in the VoIP profile used for the call to the remote gateway. The destination gateway for the call then selects the first codec of the received list, which is also included in its own VoIP profile used for that call. It will then open a media channel of that type and send a response to the initiator of the call, which only contains the selected codec. This will force the initiator of the call to also open a media channel of the same type selected on the gateway, which is the destination of the call.

Note The VoIP profile defined in the gateway will be overridden by an optional VoIP profile defined in the CS interface.

Note If no codecs are configured in the codec list of the VoIP profile, you will be unable to establish calls over the gateway.

Mode: Gateway SIP

Step	Command	Purpose
1	<code>node(gw-sip)[sip]#use profile voip profile-name</code>	Use an existing VoIP profile for this gateway.

Example: Using a VoIP profile

The following example shows how to use the existing VoIP profile myvoip for the gateway.

```
SN>enable
SN#configure
```

```
SN(cfg)#gateway sip sip
SN(gw-sip)[sip]#use profile voip myvoip
```

Binding the gateway to an IP interface

Binding the gateway to one of the available IP interfaces is required to allow the gateway to determine the local IP address it should use. The gateway needs to know the local IP address. For example, when it needs to tell the remote gateway in the call-signaling to which IP address the remote gateway should send the media streams (RTP data). The gateway always uses the IP address of the interface to which it is bound, when a local IP address is required. The following procedure describes how to bind the gateway to a local IP interface.

Mode: Gateway SIP

Step	Command	Purpose
1	<code>node(gw-hsip)[sip]#bind interface if-name</code>	Binds the gateway to the IP interface if-name.

Example: Binding the gateway

The following example shows how to bind the gateway to an IP interface.

```
SN>enable
SN#configure
SN(cfg)#gateway sip sip
SN(gw-sip)[sip]#bind interface eth0
```

Enable the Gateway

In order to become active the SIP gateway must be enabled. The following procedure enables the SIP gateway:

Mode: Gateway SIP

Step	Command	Purpose
1	<code>node(gw-sip)[sip]#no shutdown</code>	Enable the SIP gateway

Example: Enabling an SIP Gateway Configuration

The following example shows how to enable an already defined SIP Gateway.

```
SN(cfg)#gateway sip sip
SN(gw-sip)[sip]#no shutdown
```

Registering with a registrar (optional)

If necessary, the SIP gateway can register one or more users with a SIP compliant registrar. If the gateway is used as a gateway to the PSTN, this functionality is usually not required. However, if the gateway is responsible for terminals, which are directly attached to it, it is often required that the gateway registers the phone numbers or usernames associated with these terminals with a registrar.

Procedure: To register users with a SIP registrar

Mode: Gateway SIP

Step	Command	Purpose
1	<code>node(gw-sip)[sip]#registrar address [port] [use-default-server]</code>	Specify the registrar server to be used for registration. The address may be either an IP address or a name to be resolved by DNS. It is also possible to specify a SIP domain name to be resolved using DNS SRV records as the address. If the registrar uses a different UDP port than the default port 5060 for receiving messages, the port number can also be specified here. If the use-default-server parameter is specified, the REGISTER messages will be sent via the default server instead of directly to the registrar.
2	<code>node(gw-sip)[sip]#user name</code>	Adds a user to be registered with the registrar. The name can be either a name as used in sip URIs or a phone number.
3		Repeat step 2 to add more than one user to be registered.
4	<code>node(gw-sip)[sip]#registration-lifetime seconds</code>	If you want to change the lifetime of the registrations from the default of 3600 seconds to another value, you can use this command.

Example: Configuring user registration

The following example shows how to configure users to be registered on a SIP registrar.

```
SN>enable
SN#configure
SN(cfg)#gateway sip sip
SN(gw-sip)[sip]#registrar sipregistrar.mycompany.com
SN(gw-sip)[sip]#user bob
SN(gw-sip)[sip]#user 523
```

Configure a domain name (optional)

The domain name is used, if the gateway is part of a SIP domain managed by a SIP server. If configured, the gateway will use that domain name as the host part of all locally originated SIP from and to headers. If you have a SIP domain name, you can configure it on the gateway using the following procedure.

Note Do not confuse the SIP domain name with normal DNS domain names.

Note Never specify an IP address as the domain name.

Mode: Gateway SIP

Step	Command	Purpose
1	<code>node(gw-sip)[sip]#domain domain-name</code>	Define the name of your SIP domain

Example: Defining the name of the SIP domain

The following example shows how to define the SIP domain name.

```
SN(cfg)#gateway sip sip
SN(gw-sip)[sip]#domain mycompany.com
```

Configure a default server (optional)

In SIP scenarios, there is often a central SIP proxy server, to which all SIP INVITE messages must be sent. If you have such a SIP proxy server you can force the gateway to send all INVITE messages to that SIP proxy server using the following procedure.

Note You should not specify a SIP redirect server as the default server. The gateway would re-send the redirected INVITES back to the redirect server. Instead, to use a redirect server, create a SIP CS interface with the redirect server as the remote host.

Mode: Gateway SIP

Step	Command	Purpose
1	<code>node(gw-sip)[sip]#default-server address [port] {loose-router strict-router}</code>	Specify the default server to be used for all outbound INVITE messages. The address may be either an IP address or a name to be resolved by DNS. It is also possible to specify a SIP domain name to be resolved using DNS SRV records as the address. If the default server uses a different UDP port than the default port 5060 for receiving messages, the port number can also be specified here. You need also to specify, if the server is a loose-router or a strict-router. The default type is loose-router, as this is the preferred type of server in the latest SIP RFC. (RFC 3261).

Example: Configuring a default server

The following example shows how to configure a default server, which acts as a loose-router.

```
SN>enable
SN#configure
SN(cfg)#gateway sip sip
```



```
SN(gw-sip)[sip]#default server sipproxy.mycompany.com
```

Configure authentication parameters (optional)

The gateway can authenticate itself to a SIP server, if the required credentials are configured using the procedure described in this section. You can add multiple sets of authentication credentials. The gateway will use a set of authentication credentials only, if the realm of the set matches the realm of the challenge from the SIP server. The gateway always uses the set of credentials, which contains the same username as the local SIP URI of the SIP session. If there is no set of credentials, which contains the username found in the local SIP URI, the gateway will use the set of credentials for the challenged realm, which is tagged with the 'default' parameter. If no set of credentials with the 'default' parameter is available, the authentication will fail.

Mode: Gateway SIP

Step	Command	Purpose
1	<code>node(gw-sip)[sip]#authentication realm user-name password [default]</code>	Add an authentication entry for a user. Realm is the name of the authentication realm as defined on the SIP server, which requests authentication. Username and Password are the credentials to be used for authentication. The default parameter indicates that this set of credentials should also be used, if the username does not match the username of the local SIP URI. There is only one default set of credentials allowed for each realm.
2		Repeat step 1 for any additional authentication credentials you need to add.

Example: Defining authentication credentials

The following example shows how to add a set of authentication credentials for realm MySipNetwork, which uses the username Bob and the password MyPassword for any request message, for which the server requests authentication.

```
SN(cfg)#gateway sip sip
SN(gw-sip)[sip]#authentication MySipNetwork Bob MyPassword default
```

Enable 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: Gateway SIP

Step	Command	Purpose
1	<code>node(gw-sip)[sip]#session-timer seconds</code>	Enable the session timer with a refresh period of the specified number of seconds.

Example: Enabling the session timer feature

The following example shows how to enable the session timer feature, with a refresh period of 1200 seconds.

```
SN(cfg)#gateway sip sip
SN(gw-sip)[sip]#session-timer 1200
```

Advanced configuration options (optional)

Changing the listening port for inbound call-signaling

The default UDP/TCP listening port for inbound call-signaling connections is per default 5060 as defined in the SIP standard. The following procedure describes how to change the port number.

Mode: Gateway SIP

Step	Command	Purpose
1	<code>node(gw-sip)[hsip]#call-signaling-port port</code>	Defines the UDP/TCP port to use for inbound call-signaling connections

Example: Defining an alternate call-signaling port

The following example shows how to define an alternate call-signaling port on an already defined SIP Gateway.

```
SN(cfg)#gateway sip sip
SN(gw-sip)[sip]#call-signaling-port 5061
```

Define session timer version

There is currently no final standard available for the implementation of the session timer feature in SIP. However different versions of draft standard implementations are used in current SIP products. For compatibility with current SIP equipment, the gateway supports two different versions of the session-timer draft standard.

You can configure which version of the draft standard should be used. The following table shows the IETF draft standards in effect depending on the selected version.

Version	Draft standards
4	draft-ietf-sip-session-timer-04
8	draft-ietf-sip-session-timer-08

Mode: Gateway SIP

Step	Command	Purpose
1	<code>node(gw-sip)[hsip]#session-timer-version {4 8}</code>	Defines the version of the session-timer draft to be used.

Example: Defining the session-timer version

The following example shows how to use the implementation of the session timer according to draft-ietf-sip-session-timer-04.

```
SN(cfg)#gateway sip sip
SN(gw-sip)[sip]#session-timer-version 4
```

Define call transfer version

There is currently no final standard available for the implementation of the call transfer feature in SIP. However different versions of draft standard implementations are used in current SIP products. For compatibility with current SIP equipment, the gateway supports two different versions of the draft standards available for the call transfer feature.

You can configure which version of the draft standards should be used. The following table shows the IETF draft standards in effect depending on the selected version.

Version	Draft standards
2	draft-ietf-sip-cc-transfer-02
5	draft-ietf-sip-cc-transfer-05 draft-ietf-sip-refer-02 draft-ietf-sip-replaces-01

Mode: Gateway SIP

Step	Command	Purpose
1	<code>node(gw-sip)[hsip]#call-transfer-version {2 5}</code>	Defines the version of the call transfer drafts to be used.

Example: Defining the call transfer version

The following example shows how to use the implementation of the call-transfer according to draft-ietf-sip-cc-transfer-02.

```
SN(cfg)#gateway sip sip
SN(gw-sip)[sip]#call-transfer-version 2
```

Troubleshooting

You can display basic status information of the SIP gateway and the registration state of its users using the following command:

Mode: Configure

Step	Command	Purpose
1	<code>node(gw-sip)[sip]# show gateway sip status [detail]</code>	Displays SIP gateway status information. The detail parameter is a number in the range 0 to 5 and indicates how much detail should be displayed.

Example: Display SIP gateway status information

The following example shows how to display SIP gateway status information and a sample output of the command.

```
SN(cfg)#show gateway sip status
SIP Gateway: sip
=====

State:Up
User Agent:0xb69ef0
Local Address:172.16.32.19:5060
Allocated Endpoints:0
Allocated Sessions:0
Allocated Audio Datapath Controllers:0
```

The SIP gateway also provides several debugging monitors to observe the dynamic behavior of the SIP gateway. These monitors allow efficient troubleshooting of SIP problems. The most often used monitors are listed in the following table. The detail parameter is a number in the range 0 to 5 and indicates how much detail should be logged.

Command to enable the monitor	Output of the monitor
<code>node(cfg)#debug gateway sip error [detail]</code>	Logs all errors detected within the SIP gateway
<code>node(cfg)#debug gateway sip transport [detail]</code>	Logs all SIP messages sent or received over the IP network.
<code>node(cfg)#debug gateway sip registration [detail]</code>	Logs information about user registration activities
<code>node(cfg)#debug gateway sip datapath [detail]</code>	Logs information related to media channels
<code>node(cfg)#debug gateway sip signaling [detail]</code>	Logs call signaling related information

Chapter 40 **VoIP profile configuration**

Chapter contents

Introduction	476
VoIP profile configuration task list	476
Creating a VoIP profile	477
Configure codecs	478
Configuring DTMF relay	480
Configuring RTP payload types	480
Configuring the dejitter buffer (advanced)	480
Enabling/disabling filters (advanced)	483
Configuring Fax transmission	484
Configuring modem transmission	486
Examples	487
Home office in an enterprise network	487
Home office with fax	489
Soft phone client gateway	490

Introduction

This chapter gives an overview of SmartWare 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 79 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

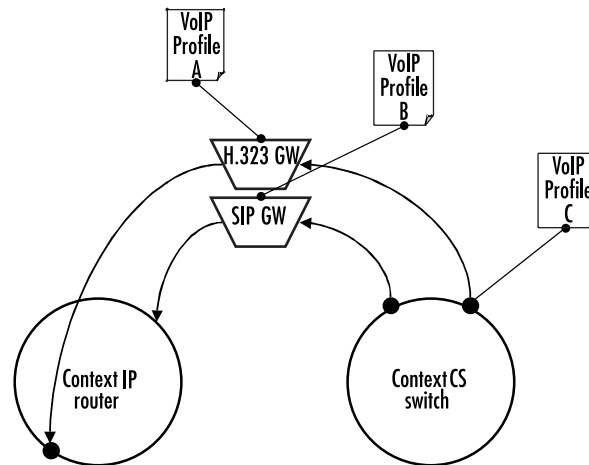


Figure 79. VoIP profile association



IMPORTANT

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

- Enabling DTMF relay (see [page 478](#))
- Configuring RTP payload types (see [page 487](#))
- Configuring the dejitter buffer (advanced) (see [page 480](#))
- Enabling/disabling filters (advanced) (see [page 483](#))
- Configuring fax transmission (see [page 484](#))
- 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	<code>node(cfg)#profile voip name</code>	Creates a VoIP profile with name <i>name</i> and enters VoIP 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 VoIP profile configuration mode is entered.
2	<code>node(pf-voip)[name]#...</code>	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 VoIP profile configuration mode.

```
SN>enable
SN#configure
SN(cfg)#profile voip g729_FaxRelay
SN(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.711 uLaw and G.711 aLaw. 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	<code>node(pf-voip)[name]#codec g729 tx-length 30 rx-length 30 silence-suppression</code>	<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.</p> <p>The following codecs are available:</p> <ul style="list-style-type: none"> • g711alaw64k • g711ulaw64k • g723-5k3 • g723-6k3 • g726-16k • g726-24k • g726-32k • g726-40k • g727-16k • g727-24k • g727-32k • g729 • netcoder-6k4 • netcoder-9k6 • transparent

Procedure: Remove a codec from the list

Mode: Profile VoIP

Step	Command	Purpose
1	<code>node(pf-voip)[name]#no codec g729</code>	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	<code>node(pf-voip)[name]#codec 1 g729 tx-length 30 rx-length 30 silence-suppression</code>	<p>Inserts codec g729 at the first position of the list (most preferred codec). The parameters are the same previously described.</p> <p>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.</p>

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 80](#). H.323 signals the DTMF tones as H.245 user input indications, SIP uses a mechanism of RTP to reliably transport tones (according to RFC2833).

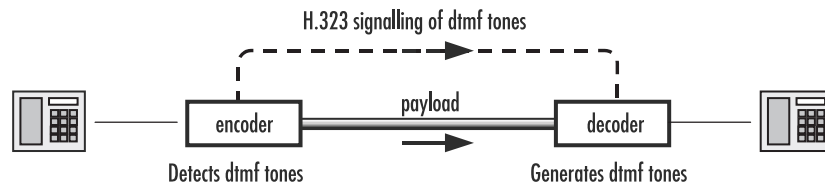


Figure 80. DTMF Relay

This procedure describes how to configure DTMF relay

Mode: Profile VoIP

Step	Command	Purpose
1	<code>node(pf-voip)[name]#dtmf-relay</code>	Enable DTMF relay

Configuring RTP payload types

If you are using DTMF relay with SIP, the DTMF digits are transported in RTP packets with a special payload type. The default value for this payload type can be configured in the profile VoIP.

Procedure: Configure RTP NTE payload type

Mode: Profile VoIP

Step	Command	Purpose
1	<code>node(pf-voip)[name]#rtp payload-type nte <i>payload-type</i></code>	Specifies the RTP payload-type for named tone events NTE (RFC2833). Default: 101.

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, SmartWare includes an integrated dejitter buffer in its 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 81](#), 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,

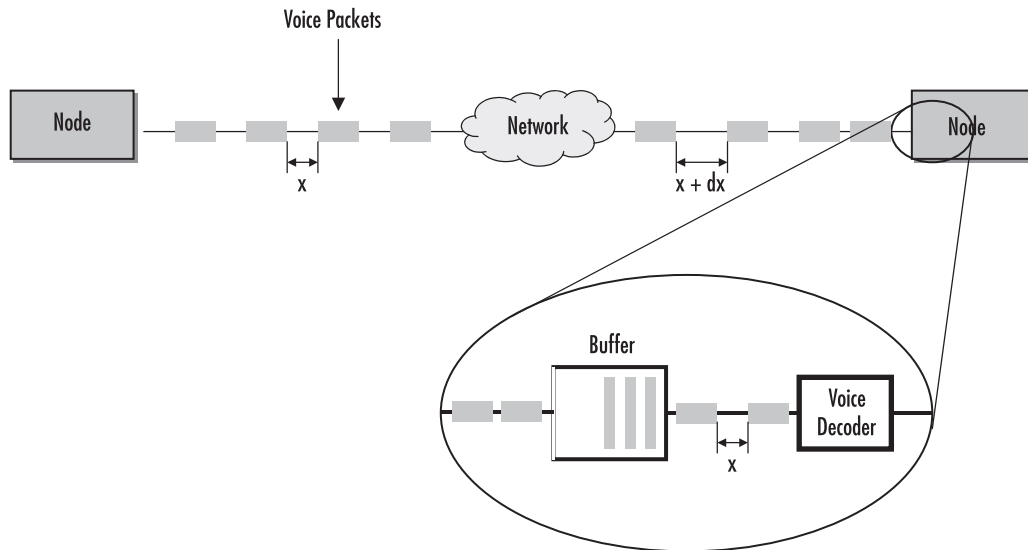


Figure 81. Jitter and de-jitter buffer

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 size is usually set to be a multiple of the expected packet arrival time in order to buffer an integral number of packets. It is not uncommon to see de-jitter buffer settings around 80 ms for each direction.

SmartWare offers the following operating modes for the dejitter buffer as illustrated in [figure 82](#):

- Adaptive—The adaptive buffer automatically adapts to variations in the network's delay characteristics and in general yields the best results for voice conversations.



IMPORTANT

In the adaptive dejitter 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

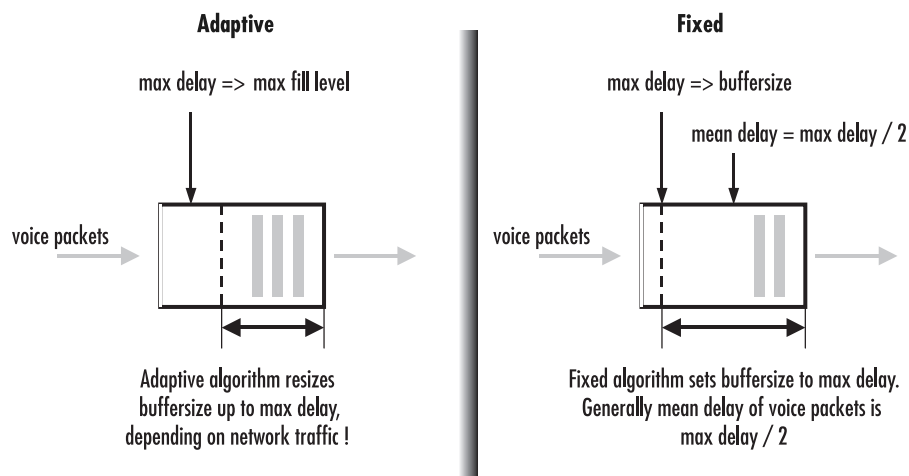


Figure 82. Adaptive versus static dejitter buffer

Procedure: Configure the dejitter buffer.

Mode: Profile VoIP

Step	Command	Purpose
1	<code>node(pf-voip)[name]#dejitte-mode mode</code>	Specify the dejitter buffer as adaptive, static or static-data.
2	<code>node(pf-voip)[name]#dejitte-max-delay max-delay</code>	Specify the maximum delay in milliseconds that the dejitter 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 SmartNodes in tandem as shown in [figure 83](#), 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.

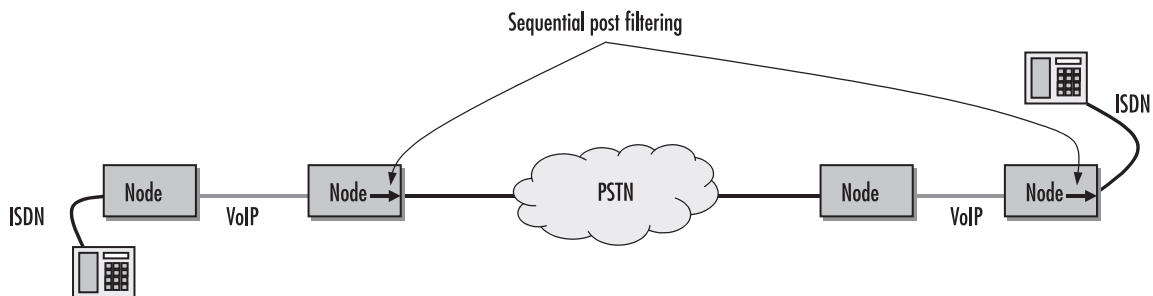


Figure 83. Multiple tandem and sequential post filtering

This procedure describes how to disable post-filtering and high-pass-filtering.

Mode: Profile VoIP

Step	Command	Purpose
1	<code>node(pf-voip)[name]#no post-filter</code>	Disable decoder output filter
2	<code>node(pf-voip)[name]#no high-pass-filter</code>	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.

```
SN>enable
SN#configure
SN(cfg)#profile voip myProfile
SN(pf-voip)[myProfi~]#no post-filter
SN(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.

SmartWare provides two solutions for fax transmission problems:

- Fax bypass—When a fax transmission is detected by the SmartNode, 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 SmartNode and sends the reference data over a fax protocol (T.38) to the receiver. Fax relay has a smaller bit-error-rate than bypass.

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 84 illustrates the difference between Fax relay and Fax bypass.

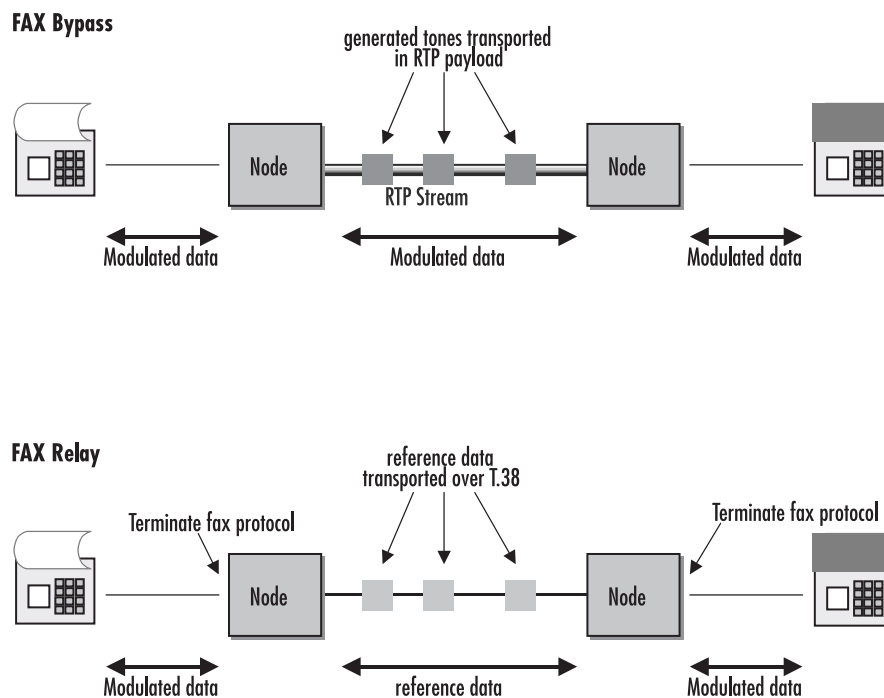


Figure 84. Fax relay and Fax bypass

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	<code>node(pf-voip)[name]#fax transmission bypass g711alaw64k</code>	Adds fax bypass transmission with codec G.711 to the list of fax transmission modes. Alternative codecs available are: <ul style="list-style-type: none"> • G.711uLaw • G.726 32kbps • G.726 24kbps.
2 optional	<code>node(pf-voip)[name]#fax dejitter-max-delay</code> <i>buffer-size</i>	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	<code>node(pf-voip)[name]# fax transmission relay t38-udp</code>	Adds fax relay transmission with T.38 protocol over UDP to the list of fax transmission modes.
2 (optional)	<code>node(pf-voip)[name]#fax redundancy ls</code> <i>low-speed-redundancy</i> hs <i>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.

Step	Command	Purpose
3 (optional)	node(pf-voip)[name]# 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.
Step 4 (optional)	node(pf-voip)[name]# 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)	node(pf-voip)[name]# fax max-bit-rate { 2400 4800 7200 9600 12000 14400 }	Sets maximum allowed bit-rate for fax relay (Default 14400 Bit/sec).
6 (optional)	node(pf-voip)[name]# 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)	node(pf-voip)[name]# 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.
8 (optional)	node(pf-voip)[name]# no fax hdlc	Disables HDLC image transfer (Default: enabled). If HDLC mode is enabled, the SmartNode 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.

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	<code>node(pf-voip)[name]#modem transmission bypass g711alaw64k</code>	Adds modem transmission with codec G.711 to the list of fax transmission modes. Alternative codecs available are: <ul style="list-style-type: none"> • G.711uLaw • G.726 32kbps • G.726 24kbps.

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 85 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.

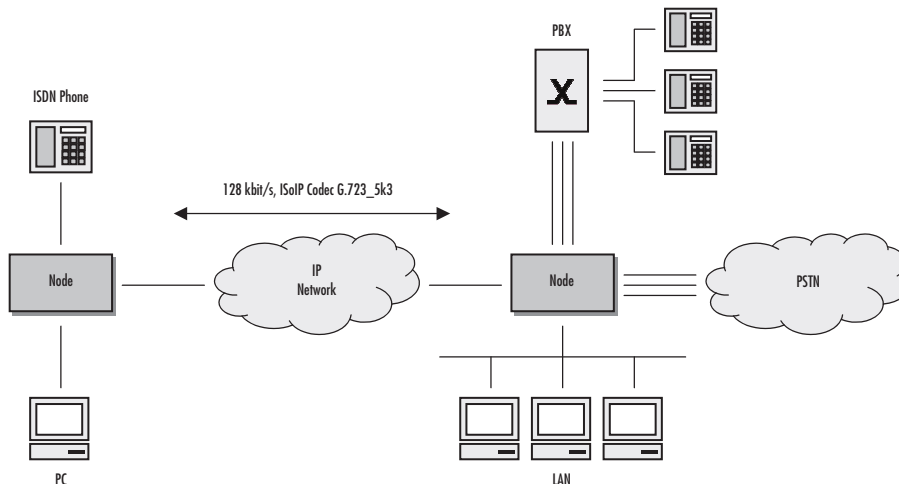


Figure 85. Home office in an enterprise network

First, configure the required CS interfaces (see chapter 27, “[CS interface configuration](#)” on page 315) and call routing (see chapter 33, “[Call router configuration](#)” on page 361).

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 SN>enable
2 SN#configure
3 SN(cfg)#profile voip Wire128kbit

```

```

4 SN(pf-voip)[Wire128~]#no codec g711aLaw64k
5 SN(pf-voip)[Wire128~]#no codec g711uLaw64k
6 SN(pf-voip)[Wire128~]#codec g723-6k3 tx-length 30 rx-length 30 silence-suppression
7 SN(pf-voip)[Wire128~]#de jitter-max-delay 100
8 SN(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
```

```
De jitter
```

```
-----
```

```
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
De jitter 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 487: 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 SN>enable
2 SN#configure
3 SN(cfg)#profile voip g729_FaxRelay
4 SN(pf-voip)[g729_Fa~]#no codec g711aLaw64k
5 SN(pf-voip)[g729_Fa~]#no codec g711uLaw64k
6 SN(pf-voip)[g729_Fa~]#codec g729 tx-length 20 rx-length 20 silence-suppression
7 SN(pf-voip)[g729_Fa~]#dejitter-max-delay 100
8 SN(pf-voip)[g729_Fa~]#fax transmission relay t38-udp
9 SN(pf-voip)[g729_Fa~]#fax max-bit-rate 9600
10 SN(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 SN>enable
```

```

2 SN#configure
3 SN(cfg)#profile voip softPhone
4 SN(pf-voip)[softPho~]#no codec g711aLaw64k
5 SN(pf-voip)[softPho~]#codec g723-6k3 tx-length 30 rx-length 30 no-silence-suppression
6 SN(pf-voip)[softPho~]#no dtmf-relay
7 SN(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
```

Description:

3. Create VoIP profile and give it a name. All settings have default values
4. Remove the default codec G.711alaw that is not supported.
5. Add codec g723-6k3 without silence-suppression
6. Disable DTMF relay.
7. Show the configured profile.

Chapter 41 **PSTN profile configuration**

Chapter contents

Introduction	494
PSTN profile configuration task list	494
Creating a PSTN profile	494
Configuring the echo canceller	495
Configuring output gain	495

Introduction

This chapter gives an overview of SmartWare 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 86 illustrates the relationship between PSTN profiles and CS interfaces. The following components are configurable:

- Echo canceller
- Output gain

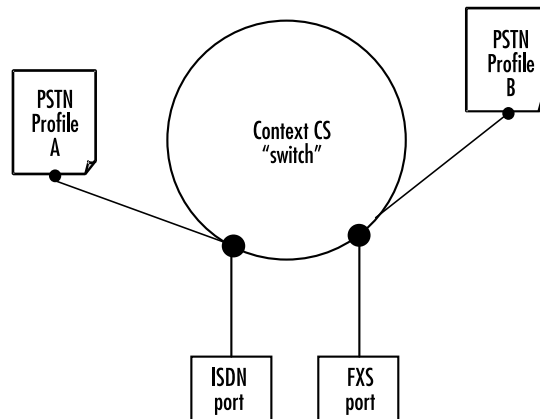


Figure 86. PSTN profile association

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 canceller (see [page 495](#))
- Configuring output gain (see [page 495](#))

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

Before configuring voice parameters, a PSTN profile must be created. 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>node(cfg)#profile pstn name</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>node(pf-pstn)[name]#...</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 87.

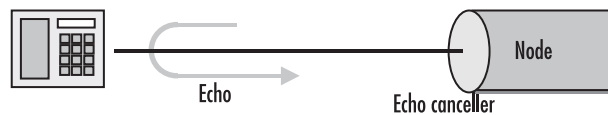


Figure 87. Echo Cancellation

Procedure: Disable echo cancellation.

Mode: Profile PSTN

Step	Command	Purpose
1	<code>node(pf-pstn)[name]#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 88.

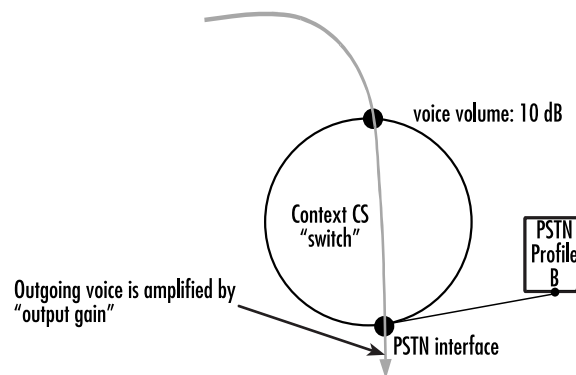


Figure 88. Applying output gain

Procedure: Configure voice output gain.

Mode: Profile PSTN

Step	Command	Purpose
1	node(pf-pstn)[name]#output-gain <i>gain</i>	Set the output gain to value in dB

Chapter 42 **VoIP debugging**

Chapter contents

Introduction	498
Debugging strategy	498
Verifying IP connectivity	499
Debugging call signaling.....	499
Debugging ISDN signaling	500
Verify an incoming call	500
Verify an outgoing call	502
Verify ISDN layer 1,2,3 status	503
Debugging FXS Signaling	504
Verify an incoming call	504
Verify an outgoing call	505
Debugging H.323 Signaling	506
Verify an incoming call	507
Verify an outgoing call	508
Debugging SIP signaling	510
Verify an incoming call	511
Verify an outgoing call	511
Using SmartWare's internal call generator	512
Debugging voice data	513
Check system logs	515
How to submit trouble reports to Patton	515

Introduction

This chapter describes how to debug VoIP sessions, including the signaling part and the voice data path part (speech, fax, and modem connectivity). 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
- Verifying IP connectivity
- Debugging call signaling (see [page 499](#))
- Debugging voice data (see [page 513](#))

Debugging strategy

Multi-service IP networks comprise highly 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. 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 monitors are switched off once the configuration is debugged and running.

Note SmartWare can create event log files that record warnings and other information from system components. Entries in the logs are time-stamped with the actual system time, so make sure the SmartNode always has the actual time as its system time. Otherwise, you are not be able to get some cleverly information from the event logs because the time stamps are always unusable.

You can enter the system time manually or have it be automatically set via SNTP. Refer to chapter 8, “[Basic system management](#)” on page 95, or chapter 20, “[SNTP client configuration](#)” on page 233.

Verifying IP connectivity

This procedure describes how to use the **ping** command to to test IP connectivity. It verifies that your SmartNode can communicate with such hosts as a gatekeeper, and IP phone, a registrar, and other VoIP gateways.

Use Telnet to access your SmartNode, then use the **ping** command to verify that an IP packet can be sent to, and received from, all hosts with which the SmartNode should be able to communicate (e.g. Gatekeeper, IP phone, Registrar, other VoIP gateways).

Mode: Administrator execution

Step	Command	Purpose
1	unit#ping <i>ipaddress</i> [<i>number-of-packets</i>] [<i>timeout seconds</i>]	Determines whether an IP host can be contacted.

Example: Verify IP connectivity

The following example uses the **ping** command to test connections to remote hosts 192.195.23.10 and 172.16.40.122. The results show an unsuccessful attempt to contact host 192.195.23.10 (as indicated by the “No route to host” response) and a successful connection to host 172.16.40.122 (which received and sent packets with no loss).

```
SN#ping 192.195.23.1 10 timeout 5
Sending 10 ICMP echo requests to 192.195.23.10, timeout is 5 seconds:
% No route to host
SN#
SN#ping 172.16.40.122
Sending 5 ICMP echo requests to 172.16.40.122, timeout is 1 seconds:
Reply from 172.16.40.122: Time <10ms
Reply from 172.16.40.122: Time <10ms
Reply from 172.16.40.122: Time <10ms
Reply from 172.16.40.122: Time <10ms
Reply from 172.16.40.122: Time <10ms
Ping statistics for 172.16.40.122:
    Packets: Sent 5, Received 5, Lost 0 (0% loss),
    RTT:      Minimum <10ms, Maximum <10ms, Average <10ms
```

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, ISDN, FXS, SIP or H.323, 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, ISDN, FXS, SIP or H.323, 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 26, “[CS context overview](#)” on page 293. 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

	Command	Purpose
	unit#debug ccisdn signaling	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.
	unit#debug ccisdn error	Prints all errors occurring in ISDN call control and ISDN datapath control. Always switch this monitor on when debugging ISDN.
	unit#debug ccisdn datapath	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.
	unit#debug isdn error	Prints all errors occurring on the ISDN port (protocol stack layers 1 to 3). Always switch this monitor on when debugging ISDN.
	unit#debug isdn event <i>slot port</i> {all layer1 layer2 layer3}	Logs in detail the operation on the ISDN port (protocol stack layers 1 to 3).
	unit#show port isdn <i>slot port status</i>	Shows the status of an ISDN port. Among others, indicates the link state of that port.
	unit#show call-control provider <i>name</i> [detail <i>detail</i>]	Shows the status of an ISDN interface in context CS (call control part of ISDN signaling). Shown are all calls currently ongoing on the interface, with their call states, signaling peers and voice data path parameters. <i>name</i> is the name of the ISDN interface. Use <i>detail</i> 5 for most verbose output.

Verify an incoming call

Make sure an incoming call is entering correctly context CS. In this example, an ISDN terminal connected to an ISDN port places a call. The port is in NET mode. The port is bound to a context CS interface named *TERMINAL*. The debug output below shows a normal (working) call setup sequence.

```
unit>enable
unit#configure
unit(cfg)#debug ccisdn error
unit(cfg)#debug ccisdn signaling
unit(cfg)#debug isdn error
unit(cfg)#18:34:10 ICC > [TERMINAL] << Message: primitive=64
18:34:10 ICC > [TERMINAL] Added endpoint TERMINAL-00b73348
18:34:10 ICC > [TERMINAL] NEW CALL. Allocated Endpoint TERMINAL-00b73348
```

```

18:34:10 ICC > [TERMINAL-00b73348] << SETUP (DSS1 Ntwk)
  Bearer capability : speech - CCITT
  circuit mode - 64kBit/s - G.711 A-law
  Calling party number : 60
  unknown number - unknown numbering plan
  presentation allowed - user provided not screened
  Called party number : 50
  unknown number - E.164 numbering plan
  High layer compatibility : telephony
  CCITT

18:34:10 ICC > [TERMINAL-00b73348] State: NULL, Event: TERMINAL SETUP IND
18:34:10 ICC > [TERMINAL-00b73348] Set state to OVERLAP SENDING
18:34:10 ICC > [TERMINAL-00b73348] >> SETUP ACKNOWLEDGEMENT (DSS1 Ntwk)

18:34:10 ICC > [TERMINAL-00b73348] State: OVERLAP SENDING, Event: PEER TRYING
18:34:10 ICC > [TERMINAL-00b73348] State: OVERLAP SENDING, Event: PEER ALERTING
18:34:10 ICC > [TERMINAL-00b73348] Set state to CALL DELIVERED
18:34:10 ICC > [TERMINAL-00b73348] >> ALERTING (DSS1 Ntwk)
  Progress indicator : inband information available
  private network serving local user - CCITT

18:34:18 ICC > [TERMINAL-00b73348] State: CALL DELIVERED, Event: PEER CONNECTED
18:34:18 ICC > [TERMINAL-00b73348] Set state to ACTIVE
18:34:18 ICC > [TERMINAL-00b73348] >> CONNECT (DSS1 Ntwk)
  Connected number : 50
  unknown number - E.164 numbering plan
  presentation allowed - user provided not screened

```

Explanation:

- The terminal places the call using en-bloc sending (no overlap dialing). In the log, this is represented by the lines “18:34:10 ICC > [TERMINAL-00b73348] << SETUP (DSS1 Ntwk) and below. This means there is a message coming from ISDN (represented by “<<” towards the left). ISDN layer1, 2 and 3 work correctly in this case.
- If the SETUP message does not appear, one of the lower ISDN layers doesn't work. Verify ISDN port status using the “show port isdn” command (see chapter 35, “ISDN port configuration” on page 425), and the “debug isdn events” commands (see below).
- The SETUP message contains different elements, among others the calling party number (60), and the called party number (50). Verify these (depending on your application, there might be as well other elements in the message).
- The line “18:34:10 ICC > [TERMINAL-00b73348] >> ALERTING (DSS1 Ntwk)” shows that the dialed number is alerting now. The ALERTING message is sent back to the terminal (represented by “>>” towards the right). You can be sure now that context CS functionality is working.
- The line “18:34:18 ICC > [TERMINAL-00b73348] >> CONNECT (DSS1 Ntwk)” indicates that the call is now established. The called party has answered the call.
- If instead of the ALERTING, a RELEASE or DISCONNECT message appears, continue debugging the outgoing call on the destination signaling protocol and the call-router.

Verify an outgoing call

Make sure a call from context CS is accepted by the connected ISDN terminal or the PSTN. In this example, an ISDN port is connected to the PSTN. The port is bound to a context CS interface named PSTN.

The simplest way to verify the signaling of an outgoing call is to use the built-in call generator. You can dial any number you know is reachable over this ISDN line. If it is the PSTN, you can dial e.g. your mobile phone number.

```

unit>enable
unit#configure
unit(cfg)#debug ccisdn error
unit(cfg)#debug ccisdn signaling
unit(cfg)#debug isdn error
unit(cfg)#call 123456 dial 987654321 dest-interface PSTN
unit(cfg)#22:03:06 ICC > [PSTN] Added endpoint PSTN-00b70a20
22:03:06 ICC > [PSTN] NEW CALL. Allocated Endpoint PSTN-00b70a20
22:03:06 ICC > [PSTN-00b70a20] >> SETUP (DSS1 User)
  Bearer capability : speech - CCITT
  circuit mode - 64kBit/s - G.711 A-law
  Calling party number : 123456
  unknown number - unknown numbering plan
  presentation allowed - user provided not screened
  Called party number : 987654321
  unknown number - unknown numbering plan
  High layer compatibility : telephony
  CCITT

22:03:06 ICC > [PSTN-00b70a20] Set state to CALL PRESENT
22:03:06 ICC > [PSTN-00b70a20] State: CALL PRESENT, Event: PEER CONNECTED
22:03:06 ICC > [PSTN] << Message: primitive=31
22:03:06 ICC > [PSTN-00b70a20] << ALERTING (DSS1 User)

22:03:06 ICC > [PSTN-00b70a20] State: CALL PRESENT, Event: PSTN ALERTING IND
22:03:06 ICC > [PSTN-00b70a20] Set state to CALL RECEIVED

unit(cfg)#call 123456 drop
unit(cfg)#22:03:14 ICC > [PSTN-00b70a20] State: CALL RECEIVED, Event: PEER
RELEASED
22:03:14 ICC > [PSTN-00b70a20] Set state to DISCONNECT INDICATION
22:03:14 ICC > [PSTN-00b70a20] >> DISCONNECT (DSS1 User)
  Cause : normal call clearing
  private network serving local user - CCITT - Q.931
  Progress indicator : inband information available
  private network serving local user - CCITT

22:03:14 ICC > [PSTN] << Message: primitive=50
22:03:14 ICC > [PSTN-00b70a20] << RELEASE COMPLETE (DSS1 User)

22:03:14 ICC > [PSTN-00b70a20] State: DISCONNECT INDICATION, Event: PSTN RELEASE
IND
22:03:14 ICC > [PSTN-00b70a20] Set state to NULL
22:03:14 ICC > [PSTN] CLEARING CALL PSTN-00b70a20
22:03:14 ICC > [PSTN] Removed endpoint PSTN-00b70a20
22:03:14 ICC > [PSTN] Destroying finished calls.
22:03:14 ICC > [PSTN] Destroyed endpoint PSTN-00b70a20

```


Explanation:

- Place a call on the command line using the line “unit(cfg)#call 123456 dial 987654321 dest-interface PSTN”. The calling party number is “123456” and the called party number “987654321”. Replace the called party with your mobile phone number, or any other number you know is reachable over the interface.
- Verify the called party number in the SETUP message on the consecutive lines.
- The line “22:03:06 ICC > [PSTN-00b70a20] << ALERTING (DSS1 User)” indicates that the called party is ringing now, and has sent back the ALERTING message to us. This means that the ISDN layer 1,2,3 and that the ISDN signaling works.
- If there is a RELEASE or DISCONNECT message instead of the ALERTING, either ISDN connectivity is not working (i.e. the message cannot be sent to the ISDN line), or the PSTN has rejected the call. Verify ISDN port status using the “show port isdn” command (see chapter 35, “ISDN port configuration” on page 425), and the “debug isdn events” commands (see below).
- Because the ALERTING is indication enough that the signaling is working, we drop the call from the command line: “unit(cfg)#call 123456 drop”
- The interface then sends a DISCONNECT message to the line, and the PSTN answers with a RELEASE COMPLETE

Verify ISDN layer 1,2,3 status

Isdn layer 1 can be verified using a “show” command:

```
EOXGW01#show port isdn 2 0 status
Proxy Isdn Driver: BRI 0 2
=====

Slot:                               2
Number of Ports:                     4
Hardware Type:                       BRI
Emergency Mode:                      off
Clock Source Port:                   0

Statistics
-----

Leased buffers:                      16
Max. leased buffers:                 30
Next Call Key:                       2

Proxy Port: BRI 0 2 0
-----

Admin State:                         Open
Real State:                          Open
Operating Layer:                     3

Layer 1
Clock Mode:                          Master
Link State:                          up
```

```

Layer 2
  Permanent Layer 2:      off
  Protocol:               PointToMultiPoint
  UniSide:                Net

Layer 3
  Protocol:               Dss1
  UniSide:                Net
  MinChannel:             0
  MaxChannel:             1
  MaxCalls:               2
  Hunt Mode:              Ascending
  Signalling Mode:        Etsi
EOXGW01#

```

The line “link state: up” tells you that layer 1 is up. To debug the layers 1-3 state machines, use the command **debug isdn event** *slot port* {all | layer1 | layer2 | layer3}.

Debugging FXS Signaling

Overview: FXS debug monitors

	Command	Purpose
	unit#debug ccfxs	Prints all operations on the FXS interfaces (high-level).
	unit#debug fxs	Prints all operations on the FXS ports (low-level).
	unit#debug voip events	Prints the dialed digits.
	unit#show call-control provider <i>name</i> [detail <i>detail</i>]	Shows the status of an FXS interface in context CS (call control part of FXS signaling). Shown are all calls currently ongoing on the interface, with their call states, signaling peers and voice data path parameters. <i>name</i> is the name of the FXS interface. Use <i>detail 5</i> for most verbose output.

Verify an incoming call

Make sure an incoming call is entering correctly context CS. In this example, a POTS terminal connected to an FXS port goes off hook and places a call. The port is bound to a context CS interface named PHONE. The debug output below shows a normal (working) call setup sequence.

```

unit>enable
unit#configure
unit(cfg)#debug ccfxs
unit(cfg)#debug fxs
unit(cfg)#09:00:11 FXS > [0 1] Off hook.
09:00:11 FXS > [0 1] Notifying off hook.
09:00:11 FXS > FXS-00/01: state=on-hook, event=off-hook
09:00:11 FXS > [0 1] Set state to 'Active'
09:00:11 FXS > FXS-00/01: new state=filtering-events
09:00:11 CFXS > [EP PHONE] Change state to DIAL-TONE.
09:00:11 CFXS > [EP PHONE] Change datapath direction to receive-only.

```

```

09:00:11 CFXS > [EP PHONE] Play tone: dial-tone
09:00:11 FXS > FXS-00/01: state=filtering-events, event=timeout
09:00:11 FXS > FXS-00/01: new state=off-hook
09:00:13 CFXS > [EP PHONE] Change state to DIALING.
09:00:13 CFXS > [EP PHONE] Stop tone.
09:00:43 CFXS > [EP PHONE] Play tone: ringback-tone
09:01:01 CFXS > [EP PHONE] Change state to CONNECTED.
09:01:01 CFXS > [EP PHONE] Stop tone.
09:01:01 CFXS > [EP PHONE] Change datapath direction to send/receive.

```

Explanation:

- “unit(cfg)#09:00:11 FXS > [0 1] Off hook.“: The phone went off-hook
- “09:00:11 CFXS > [EP PHONE] Change state to DIAL-TONE.“: The CS interface has The dial-tone is played. This indicates that the binding between
- “09:00:13 CFXS > [EP PHONE] Change state to DIALING.“: The first digit has been touched. Dial-tone stops.
- “09:00:43 CFXS > [EP PHONE] Play tone: ringback-tone“: The called-party is ringing, alerting-tone is played to the phone. This means that the call arrived correctly in context CS, and has reached its destination.
- If the above line does not appear, the call has arrived in context CS, but has not reached its destination. Continue debugging call-router and the outgoing call to the destination signaling protocol.
- If you want to see the dialed digits in the debug monitor, use “debug voip events”.
- “09:01:01 CFXS > [EP PHONE] Change state to CONNECTED.“: the called party has accepted the call, and the call is now established.

Verify an outgoing call

Make sure a call from context CS is accepted by the connected POTS terminal. The port is bound to a context CS interface named PHONE.

The simplest way to verify the signaling of an outgoing call is to use the built-in call generator. You can place a call to any called-party, and verify that the connected phone is ringing.

```

unit>enable
unit#configure
unit(cfg)#debug ccfxs
unit(cfg)#debug fxs
unit(cfg)#call 1 dial 2 dest-interface PHONE
unit(cfg)#09:13:55 CFXS > [EP PHONE] Change state to RINGING.
09:13:55 CFXS > [EP PHONE] Start Ring.
09:13:55 FXS > FXS-00/01: state=on-hook, event=ring-start
09:13:55 FXS > [0 1] Set state to 'Ringing'
09:13:55 FXS > FXS-00/01: new state=ringing
09:13:56 FXS > [0 1] Set state to 'RingPause'
09:14:00 FXS > [0 1] Set state to 'Ringing'
09:14:01 FXS > [0 1] Set state to 'RingPause'
09:14:02 FXS > [0 1] Off hook.
09:14:02 FXS > [0 1] Notifying off hook.
09:14:02 FXS > FXS-00/01: state=ringing, event=off-hook
09:14:02 FXS > [0 1] Set state to 'Idle'

```

```

09:14:02 FXS > [0 1] Set state to 'Active'
09:14:02 FXS > FXS-00/01: new state=off-hook
09:14:02 CFXS > [EP PHONE] Change state to CONNECTED.
09:14:02 CFXS > [EP PHONE] Stop ring.
09:14:02 FXS > FXS-00/01: state=off-hook, event=ring-stop
09:14:02 FXS > FXS-00/01: new state=off-hook
09:14:02 CFXS > [EP PHONE] Change datapath direction to send/receive.

```

Explanation:

- “unit(cfg)#09:13:55 CFXS > [EP PHONE] Change state to RINGING.“: The context CS interface changes the state to RINGING, that means it has accepted the call.
- “09:13:55 FXS > [0 1] Set state to 'Ringing'“: The port begins to ring on the analog line. This means that the binding of the port to the context CS interface is correct, and your phone should now be ringing.
- If the phone does not start to ring, revise the binding of the port to the context CS interface, and the connectors of the POTS line.
- “09:14:02 CFXS > [EP PHONE] Change state to CONNECTED.“: The phone went off-hook and thus accepted the call. The state of the CS interface goes to CONNECTED. The call is now established.

Debugging H.323 Signaling

Overview: H.323 debug monitors

	Command	Purpose
	unit#debug gateway h323 signaling	Prints all signaling operations on H.323 interfaces.
	unit#debug gateway h323 error	General purpose error monitor of H.323. Always enable this when debugging H.323.
	unit#debug gateway h323 datapath	Prints operations on the H.323 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.
	unit#show gateway h323 status	Shows the state of the H.323 gateway. Among others: enabled/disabled, RAS registration state.
	unit#show gateway h323 call <i>name</i> [detail <i>detail</i>]	Shows all calls ongoing on the H.323 interface on context CS with name <i>name</i> .
	unit#show call-control provider <i>name</i> [detail <i>detail</i>]	Shows the status of an H.323 interface in context CS (call control part of H.323 signaling). Shown are all calls currently ongoing on the interface, with their call states, signaling peers and voice data path parameters. <i>name</i> is the name of the H.323 interface. Use <i>detail 5</i> for most verbose output.

See also section “[Troubleshooting](#)” on page 463 for further informations on debugging H.323. There are more than 40 debug monitors, such as 'q931', 'ras' or 'signaling'. For a list of all available debug monitors, refer to the CLI online help.

Verify an incoming call

Make sure an incoming call is entering correctly context CS. In this example, a VoIP gateway or VoIP phone on the network makes a call to your unit using a gatekeeper. The debug output below shows a normal (working) call setup sequence

```

unit#enable
unit#configure
unit(cfg)#debug gateway h323 error
unit(cfg)#debug gateway h323 signaling
unit#debug gateway h323 ras
unit(cfg)#show gateway h323 status
H.323 Gateway: h323
=====

State:                               UP
Stack Handle:                         0x193ce44

RAS Engine
-----

State:                               REGISTERED
Gatekeeper:                           172.16.32.51/1719

Allocated Endpoints:                  0
Allocated RAS Engines:                 1
Allocated Control Channels:            0
Allocated Outgoing Logical Slowstart Channels: 0
Allocated Outgoing Logical Faststart Channels: 0
Allocated Incoming Logical Channels:   0
unit#
unit#00:29:03 H323 > [EP h323-00c13dc0] Stack: Allocated new call: 0x00be56b0
00:29:03 H323 > Provider SN_33: Added endpoint h323-00c13dc0
00:29:03 H323 > [EP h323-00c13dc0] Stack: Dial to remote terminal
00:29:03 H323 > [EP h323-00c13dc0] Destination Address:           TEL:60,60
00:29:03 H323 > [EP h323-00c13dc0] Source Address:                TEL:50,50
00:29:03 H323 > [EP h323-00c13dc0] Presentation Indicator:      Presentation
allowed
00:29:03 H323 > [EP h323-00c13dc0] Screening Indicator:          User pro-
vided, not screened
00:29:03 H323 > [EP h323-00c13dc0] Information Transfer Capability: Speech
00:29:03 H323 > [EP h323-00c13dc0] Display:                       50
00:29:03 H323 > [EP h323-00c13dc0] User-User:
00:29:03 H323 > [EP h323-00c13dc0] Set state to TERMINAL TRYING
00:29:03 H323 > [EP h323-00c13dc0] State: TERMINAL TRYING, Call Event: PEER CON-
NECTED
00:29:03 H323 > [EP h323-00c13dc0] State: TERMINAL TRYING, Call Event: PROGRESS
00:29:03 HRAS > Stack: Received Admission Confirm
00:29:03 H323 > [EP h323-00c13dc0] Stack: State: DIALTONE
00:29:03 H323 > [EP h323-00c13dc0] Call-ID is
0213:4d80:each:11e0:2eee:0030:2b00:1e0e
00:29:03 H323 > [EP h323-00c13dc0] Stack: Received Incomplete-Address Indication
00:29:03 H323 > [EP h323-00c13dc0] Stack: State: RINGBACK
00:29:03 H323 > [EP h323-00c13dc0] Stack: Received ALERTING
00:29:03 H323 > [EP h323-00c13dc0] Progress Indicator: (none)

```

```

00:29:03 H323 > [EP h323-00c13dc0] State: TERMINAL TRYING, Call Event: TERMINAL
ALERTING
00:29:03 H323 > [EP h323-00c13dc0] Set state to TERMINAL ALERTING
00:29:08 H323 > [EP h323-00c13dc0] Stack: State: CONNECTED (CALL-SETUP)
00:29:08 H323 > [EP h323-00c13dc0] State: TERMINAL ALERTING, Call Event: TERMINAL
CONNECTED
00:29:08 H323 > [EP h323-00c13dc0] Set state to CONNECTED
00:29:08 H323 > [EP h323-00c13dc0] Stack: Send STATUS INQUIRY
00:29:08 H323 > [EP h323-00c13dc0] Stack: Opening the H.245 control-channel
00:29:08 H323 > [EP h323-00c13dc0] State: CONNECTED, Call Event: PROGRESS
00:29:08 H323 > [EP h323-00c13dc0] Stack: Received STATUS. Audit successful
00:29:08 H323 > [EP h323-00c13dc0] Channel State: IDLE, Channel Event: CONTROL-UP
00:29:08 H323 > [EP h323-00c13dc0] Set channel state to OPENING
00:29:08 H323 > [EP h323-00c13dc0] State: CONNECTED, Call Event: PROGRESS
00:29:08 H323 > [EP h323-00c13dc0] Stack: State: CONNECTED (CALL)
00:29:08 H323 > [EP h323-00c13dc0] Stack: New Incoming Logical Channel: 00bcc448
00:29:08 H323 > [EP h323-00c13dc0] Channel State: OPENING, Channel Event: MODE-UP
00:29:08 H323 > [EP h323-00c13dc0] Set channel state to UP

```

Explanation:

- First the state of the gateway is checked. The state is “UP”, and the RAS engine is “REGISTERED”, which is OK.
- The line “00:29:03 H323 > [EP h323-00c13dc0] Stack: Allocated new call: 0x00be56b” tells that there is a new call incoming from H.323. This means that transport layer is OK. If there is no debug output at all, try to use “debug gateway h323 tpktchan”, which monitors all H.323 socket TCP traffic.
- The line “00:29:03 H323 > [EP h323-00c13dc0] Destination Address: TEL:60,60” and below show the calling and called party properties. In the present case, the called party number is 60.
- “00:29:03 HRAS > Stack: Received Admission Confirm” says that the gatekeeper allowed us to accept the call. Continue debugging call routing if the call does not reach its destination.

Verify an outgoing call

Make sure an outgoing call leaves correctly on text CS and the device. In this example, a terminal on the gateway to debug wants to make a call towards a H.323 VoIP network using a gatekeeper. The debug output below shows a normal (working) call setup sequence

```

unit#enable
unit#configure
unit(cfg)#debug gateway h323 error
unit(cfg)#debug gateway h323 signaling
unit#debug gateway h323 ras
unit(cfg)#show gateway h323 status
H.323 Gateway: h323
=====

State:                               UP
Stack Handle:                         0x193ce44

RAS Engine
-----

```

```

State: REGISTERED
Gatekeeper: 172.16.32.51/1719

Allocated Endpoints: 0
Allocated RAS Engines: 1
Allocated Control Channels: 0
Allocated Outgoing Logical Slowstart Channels: 0
Allocated Outgoing Logical Faststart Channels: 0
Allocated Incoming Logical Channels: 0
unit#
unit#01:00:10 H323 > [EP h323-00c07230] Stack: Allocated new call: 0x00be5968
01:00:10 H323 > Provider SN_33: Added endpoint h323-00c07230
01:00:10 H323 > [EP h323-00c07230] Stack: Dial to remote terminal
01:00:10 H323 > [EP h323-00c07230] Destination Address: TEL:60,60
01:00:10 H323 > [EP h323-00c07230] Source Address: TEL:50,50
01:00:10 H323 > [EP h323-00c07230] Presentation Indicator: Presentation
allowed
01:00:10 H323 > [EP h323-00c07230] Screening Indicator: User pro-
vided, not screened
01:00:10 H323 > [EP h323-00c07230] Information Transfer Capability: Speech
01:00:10 H323 > [EP h323-00c07230] Display: 50
01:00:10 H323 > [EP h323-00c07230] User-User:
01:00:10 H323 > [EP h323-00c07230] Set state to TERMINAL TRYING
01:00:10 H323 > [EP h323-00c07230] State: TERMINAL TRYING, Call Event: PEER CON-
NECTED
01:00:10 H323 > [EP h323-00c07230] State: TERMINAL TRYING, Call Event: PROGRESS
01:00:10 HRAS > Stack: Received Admission Confirm
01:00:10 H323 > [EP h323-00c07230] Stack: State: DIALTONE
01:00:10 H323 > [EP h323-00c07230] Call-ID is
0213:4d8d:13ba:1db8:2eef:0030:2b00:1e0e
01:00:11 H323 > [EP h323-00c07230] Stack: Received Incomplete-Address Indication
01:00:11 H323 > [EP h323-00c07230] Stack: State: RINGBACK
01:00:11 H323 > [EP h323-00c07230] Stack: Received ALERTING
01:00:11 H323 > [EP h323-00c07230] Progress Indicator: (none)
01:00:11 H323 > [EP h323-00c07230] State: TERMINAL TRYING, Call Event: TERMINAL
ALERTING
01:00:11 H323 > [EP h323-00c07230] Set state to TERMINAL ALERTING
01:00:14 H323 > [EP h323-00c07230] Stack: State: CONNECTED (CALL-SETUP)
01:00:14 H323 > [EP h323-00c07230] State: TERMINAL ALERTING, Call Event: TERMINAL
CONNECTED
01:00:14 H323 > [EP h323-00c07230] Set state to CONNECTED
01:00:14 H323 > [EP h323-00c07230] Stack: Send STATUS INQUIRY
01:00:14 H323 > [EP h323-00c07230] Stack: Opening the H.245 control-channel
01:00:14 H323 > [EP h323-00c07230] State: CONNECTED, Call Event: PROGRESS
01:00:14 H323 > [EP h323-00c07230] Stack: Received STATUS. Audit successful
01:00:14 H323 > [EP h323-00c07230] Channel State: IDLE, Channel Event: CONTROL-UP
01:00:14 H323 > [EP h323-00c07230] Set channel state to OPENING
01:00:14 H323 > [EP h323-00c07230] State: CONNECTED, Call Event: PROGRESS
01:00:14 H323 > [EP h323-00c07230] Stack: State: CONNECTED (CALL)
01:00:14 H323 > [EP h323-00c07230] Stack: New Incoming Logical Channel: 00bccd68
01:00:14 H323 > [EP h323-00c07230] Channel State: OPENING, Channel Event: MODE-UP
01:00:14 H323 > [EP h323-00c07230] Set channel state to UP

```

Explanation:

- First the state of the gateway is checked. The state is “UP”, and the RAS engine is “REGISTERED”, which is OK.
- The line “01:00:10 H323 > [EP h323-00c07230] Stack: Allocated new call: 0x00be5968“ tells that there is a new call incoming from H.323. This means that transport layer is OK. If there is no debug output at all, try to use “debug gateway h323 tpkchan”, which monitors all H.323 socket TCP traffic.
- The line “01:00:10 H323 > [EP h323-00c07230] Destination Address: TEL:60,60“ and below show the calling and called party properties. In the present case, the called party number is 60.
- “00:29:03 HRAS > Stack: Received Admission Confirm“ says that the gatekeeper allowed us to dial to H.323 network with the given call parameters. If you can't see any H.323 SETUP message on the net, or if the called gateway does not indicate any H.323 message received, use “debug gateway h323 tpkchan” to verify that a packet is being sent.

Debugging SIP signaling

Overview: SIPdebug monitors

	Command	Purpose
	unit#debug gateway sip transport [detail <i>detail</i>]	Prints all SIP messages that are sent / received. This is a good monitor to start with when debugging SIP. If <i>detail</i> is increased (up to 5), more verbose message content is printed.
	unit#debug gateway sip error	General purpose error monitor of SIP. Always enable this when debugging SIP.
	unit#debug gateway sip registration	Prints all actions concerning SIP registration mechanism.
	unit#debug gateway sip signaling	Prints all signaling operations on SIP interfaces.
	unit#debug gateway sip datapath	Prints operations on the SIP 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.
	unit#show gateway sip status	Shows the state of the SIP gateway. Among others: enabled/disabled, RAS registration state.
	unit#show gateway sip call <i>name</i> [detail <i>detail</i>]	Shows all calls ongoing on the SIP interface on context CS with name <i>name</i> .
	unit#show call-control provider <i>name</i> [detail <i>detail</i>]	Shows the status of an SIP interface in context CS (call control part of SIP signaling). Shown are all calls currently ongoing on the interface, with their call states, signaling peers and voice data path parameters. <i>name</i> is the name of the SIP interface. Use <i>detail 5</i> for most verbose output.

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 gateway sip error
unit(cfg)#debug gateway 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 from context CS leaves correctly to the SIP network. The following sequence shows a working call setup.

```
unit(cfg)#debug gateway sip error
unit(cfg)#debug gateway 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.

Using SmartWare's internal call generator

The SmartWare 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.

call <i>calling-party</i> { accept alerting dial drop inband-info offer-state proceeding reject-all resume suspend user-input } ...	Manipulates calls locally generated from the center of context CS. To create a call, use "call <i>calling-party</i> dial <i>called-party</i> ", to manipulate this call afterwards use the same calling party. See the examples below for usage.
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Example: Debug ISDN protocol using the call generator. Create a call from context CS to an ISDN interface called TERMINAL.

```

unit(cfg)#debug ccisdn signaling
unit(cfg)#debug ccisdn error
unit(cfg)#debug isdn error
unit(cfg)#call 55 dial 50 dest-interface TERMINAL
unit(cfg)#19:17:38 ICC > [TERMINAL] Added endpoint TERMINAL-00df2760
19:17:38 ICC > [TERMINAL] NEW CALL. Allocated Endpoint TERMINAL-00df2760
19:17:38 ICC > [TERMINAL-00df2760] >> SETUP (DSS1 Ntwk)
  Bearer capability : speech - CCITT
  circuit mode - 64kBit/s - G.711 A-law
  Calling party number : 55
  unknown number - unknown numbering plan
  presentation allowed - user provided not screened
  Called party number : 50
  unknown number - unknown numbering plan
  High layer compatibility : telephony
  CCITT

19:17:38 ICC > [TERMINAL-00df2760] Set state to CALL PRESENT
19:17:38 ICC > [TERMINAL-00df2760] State: CALL PRESENT, Event: PEER CONNECTED
19:17:38 ICC > [TERMINAL] << Message: primitive=31
19:17:38 ICC > [TERMINAL-00df2760] << ALERTING (DSS1 Ntwk)

19:17:38 ICC > [TERMINAL-00df2760] State: CALL PRESENT, Event: TERMINAL ALERTING
IND
19:17:38 ICC > [TERMINAL-00df2760] Set state to CALL RECEIVED
19:17:44 ICC > [TERMINAL] << Message: primitive=33
19:17:44 ICC > [TERMINAL-00df2760] << CONNECT (DSS1 Ntwk)
  Connected number : 50
  unknown number - unknown numbering plan
  presentation allowed - user provided not screened

19:17:44 ICC > [TERMINAL-00df2760] State: CALL RECEIVED, Event: TERMINAL CONNECT
IND
19:17:44 ICC > [TERMINAL-00df2760] Set state to ACTIVE
19:17:44 ICC > [TERMINAL-00df2760] >> CONNECT ACKNOWLEDGEMENT (DSS1 Ntwk)
unit(cfg)#
unit(cfg)#call 55 drop
unit(cfg)#19:19:29 ICC > [TERMINAL-00df2760] State: ACTIVE, Event: PEER RELEASED
19:19:29 ICC > [TERMINAL-00df2760] Set state to DISCONNECT INDICATION
19:19:29 ICC > [TERMINAL-00df2760] >> DISCONNECT (DSS1 Ntwk)

```

```
Cause : normal call clearing
      private network serving local user - CCITT - Q.931
Progress indicator : inband information available
      private network serving local user - CCIT
```

Explanation:

- “unit(cfg)#call 55 dial 50 dest-interface TERMINAL“: Dial the number 50, with calling-party 55 to the interface TERMINAL. If a phone connected to the port that binds to interface TERMINAL has MSN 50 configured, it will start to ring. This is the case in our example.
- You just have verified that ISDN layer1-3 and the context CS interface TERMINAL are configured so that a call with called-party 50 can be handled. If this is not the case (no response or a RELEASE message), continue debugging ISDN signaling.
- “unit(cfg)#call 55 drop“: Drops the call initiated with the dial command.

You can proceed as in this example with any other context CS interface, also for VoIP protocols like SIP and H.323.

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.

An overview of all available VoIP debug monitors is given below. Some more specific examples for debugging cases follow after that.

Overview: VoIP and DSP debug monitors

debug voip dejitter	Displays changes to the settings of the dejitter buffer, exceptions (under-run, overrun, packet drops) and size changes. Usage: To investigate problems related to voice quality, voice packet payload sizes, delays, jitter
debug voip events	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.
debug voip rtp	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).
debug voip t38 events	Displays T.38 related call parameters at call setup, and operational errors like lost packets. Usage: To investigate problems with T.38 Fax transmissions in general

debug voip t38 dejitter	Tracks the operation of a special dejitter buffer used for T.38 Fax transmissions. Usage: To document the influence of network quality on T.38 Fax transmissions.
debug voip demux	Displays changes to the "context demultiplexer". Usage: To identify problems when no voice connection can be established.
debug voip cs	Displays control activities on the TDM part of the Data Path: DSP resource allocation, TDM timeslot switching, hair pinning). Usage: To investigate problems with hair pinning or timeslot switching.
debug dsp events	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.
debug dsp errors	Displays DSP errors (e.g. configuration errors, DSP software crashes, under/overruns). Usage: To investigate in all kind of problems that involve a DSP (voice connections, fax, tones, ...)
debug dsp t30	Traces the flow of T.30 communication states between the fax machines (for T.38 Fax transmission only), and logs changes. Gives approx. 10lines of debug output per transmitted fax page. Usage: To debug all T.38 Fax related problems.
debug dsp ifp	Decodes the IFP (Internet Fax Protocol) elements within T.38 packets. More detailed decoding than T.30 monitor, but can generate a lot of output. usage: To debug Fax related problems.

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 (see chapters Debug H.323 Data and Debug Session Control Data).

Use the following debug monitors to start with:

Step	Command	Purpose
1	unit#debug voip events	Enable the voip events monitor .
2	unit#debug dsp error	Enable the DSP error monitor.
3	unit#debug voip rtp	Enable the RTP debug monitor

Depending on the type of problem, continue using one or more of the following monitors:

Command	Purpose
unit#debug dsp events	Enable the DSP events monitor .
unit#debug voip dejitter	Enable the dejitter buffer monitor.
unit#debug voip rtp	Enable the RTP debug monitor
unit#debug voip cs	Enable the CS debug 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 voice data" on page 513).

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 H.323 is faulty. The session-control and H.323 signaling monitors (see Debug Session Control Data, and Debug H.323 Data) might also be helpful to identify these cases.

Use the following debug monitors to start with, if you assume that signaling is OK:

Step	Command	Purpose
1	unit#debug voip events	Enable the voip events monitor .
2	unit#debug voip t38 events	Enable the T.38 events monitor
3	unit#debug voip t38 dejitter	Enable the T.38 dejitter monitor
4	unit#debug dsp error	Enable the DSP error monitor.
5	unit#debug dsp t30	Enable the T.30 flow monitor

Depending on the type of problem, use also:

Command	Purpose
unit#debug dsp events	Enable the DSP event monitor
unit#debug dsp ifp	Enable the IFP decoding monitor

Check system logs

See section "Displaying the system logs" on page 102.

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)
- **Your location**—For further enquiries please add your email address and phone number.

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 (www.sniffer.com)

TTC Firebird (www.ttc.com)

Ethereal (freeware; www.ethereal.com)

When possible, submit the package of trouble report files by email to the following address: support@patton.com (use fax only in exceptional cases).

Appendix A **Terms and definitions**

Chapter contents

Introduction	518
SmartWare architecture terms and definitions	518

Introduction

This chapter contains the terms and their definitions that are used throughout this *SmartWare Software Configuration 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, such as the integrated services digital network (ISDN), public switched telephone network (PSTN), and plain old telephone service (POTS).

SmartWare architecture terms and definitions

Term or Definition	Meaning
Administrator	The person who has privileged access to the SmartWare CLI.
Application Download	A application image is downloaded from a remote TFTP server to the persistent memory (flash:) of a SmartNode.
Application Image	The binary code of the SmartWare stored in the persistent memory (flash:) of a SmartNode.
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 SmartNode.
Bootloader	The bootloader is a “mini” application performing basic system checks and starting the SmartWare application. The bootloader also provides minimal network services allowing the SmartNode to be accessed and upgraded over the network even if the SmartWare application 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 SmartNode.
Bootstrap	The starting-up of a SmartNode, 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 bootloader.
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 SmartNode 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 SmartNode 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 SmartWare 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 SmartNode.
Configuration File	The configuration file contains the SmartWare CLI commands, which are used to configure the software modules of the SmartWare performing a certain functionality of the SmartNode.
Configuration Server	A central server used as a store for configuration files, which are downloaded to or uploaded from a SmartNode using TFTP.
Configuration Upload	A configuration file is uploaded from the persistent memory (nvram:) or volatile memory (system:) of a SmartNode via TFTP to a TFTP server.
Context	A SmartWare 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	To compensate variable network delays, the SmartWare includes a dejitter buffer. Storing packets in a dejitter buffer before they are transferred to the local ISDN equipment, e.g. telephone, the SmartWare converts a variable delay into a fixed delay, giving voice a better quality. See also <i>Jitter</i> .
Digit Collection	SmartWare supports overlap dialing. Some of the connected devices (PBX, ISDN network, remote gateways and gatekeepers) may however require bloc sending of the dialed number. The SmartWare 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 SmartNode.
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 SmartNode.
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 SmartNode.
Fast Connect	A "normal" call setup with H.323 requires several TCP segments to be transmitted, because various parameters are negotiated. 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 SmartNode containing the Application Image, Bootloader Image and the driver software Image.

Term or Definition	Meaning
flash:	A region in the persistent memory of a SmartNode. See also <i>flash memory</i> .
Gatekeeper	Gatekeepers manage H.323 zones, which are logical collections of devices such as all H.323 devices within an IP subnet. For example, gatekeepers provide address translation (routing) for the devices in their zone.
Gateway	In SmartWare terminology 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. SmartWare includes an H.323 and SIP gateway.
H.323	ITU-T recommendation H.323 describes terminals, equipment and services for multimedia communication over Local Area Networks (LAN) which do not provide a guaranteed quality of service. H.323 terminals and equipment may carry real-time voice, data and video, or any combination, including video telephony.
H.323 RAS	H.323 registration authentication service (RAS) is a sub protocol of H.323. The RAS signaling protocol performs registration, admissions, and bandwidth changes and disengage procedures between the VoIP gateway and the gatekeeper.
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 SmartNode 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	In SmartWare 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 SmartNode 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 SmartNode. See also <i>de jitter buffer</i> .
Mode	The SmartWare CLI is comprised of modes. There are two basic mode groups, the execution mode group and the configuration mode group.

Term or Definition	Meaning
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 SmartNode. 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 SmartNode or network element.
nvram:	Persistent memory section of a SmartNode containing the startup configuration, the factory configuration and used defined configurations.
Operator	The person who has limited access to the SmartWare CLI.
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.
PMC	The optional interface cards for SmartNode 2000 series which are compatible to the PCI Mezzanine Card standards.
PMC Driver Software	PMC driver software performs the runtime tasks on the PMC interface card mounted in SmartNode 2000 series devices. The PMC drivers are interface card specific and also have build numbers. Refer to the SmartWare release notes for PMC driver software compatibility. The PMC drivers may be upgraded together with the SmartWare release or they can be downloaded individually into the persistent memory (flash:) of a SmartNode.
PMC Loader	The PMC loader initializes the PMC interface card mounted in SmartNode 2000 series of devices. It checks hardware versions and determines if compatible PMC drivers are available. The PMC loader may be upgraded together with the SmartWare release.
Port	In SmartWare a port represents a physical connector on the SmartNode.
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.

Term or Definition	Meaning
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	SmartWare is organized in releases that define the main voice and data features of a SmartNode. Several builds can be available from certain release. See also <i>build</i> .
Routing Engine	In SmartWare the routing engine handles the basic IP routing.
Running Configuration	The currently running configuration (running-config) for SmartWare, which is executed from the volatile memory (system:) on the SmartNode.
SmartNode	<p>The SmartNode is Patton Electronics' networking product available in three series:</p> <ul style="list-style-type: none"> • The SmartNode 1000 series are compact integrated access devices for applications in SOHO or branch office environments. They are available in a various interface configurations supporting up to 4 voice channels • The SmartNode 2000 series are modular integrated services network nodes designed for medium and large enterprise applications. Multiple PMC based interface slots and a range of interface cards provides flexibility for both LAN and WAN interface configuration. • The SmartNode 4000 series are compact integrated access and VoIP gateway devices for applications in SOHO or branch office environments for a seamless integration of analog telephony equipment into VoIP solutions. They are available in various interface configurations supporting up to 8 x FXS or FXO.
SmartWare	SmartWare is the application software running on the SmartNode hardware platforms. The SmartWare is available in several releases that in general support all currently available SmartNode models.
SmartView Management Center	SmartView Management Center is a suite of element and network management applications that enable the management integration of the SmartNode platforms in a provider service and network management system. SmartView Management Center ensures efficient operations for SmartNode networks growing in size and complexity.
Session Router	Calls through SmartNode 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 SmartNode 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.

Term or Definition	Meaning
System Memory	The volatile memory, that includes the system: region, holding the running-config for the SmartWare during operation of a SmartNode.
system:	A region in the volatile memory of a SmartNode. See also <i>system memory</i> .
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 **Mode summary**

Chapter contents

Introduction.....	526
-------------------	-----

Introduction

Figure 89 on page 526, figure 90 on page 527, and figure 91 on page 528 show the configuration mode hierarchy. Each box contains the mode name, the command to enter in this mode and the mode prompt printed in a Telnet or console session. The commands are defined in appendix C, “Command summary” on page 529.

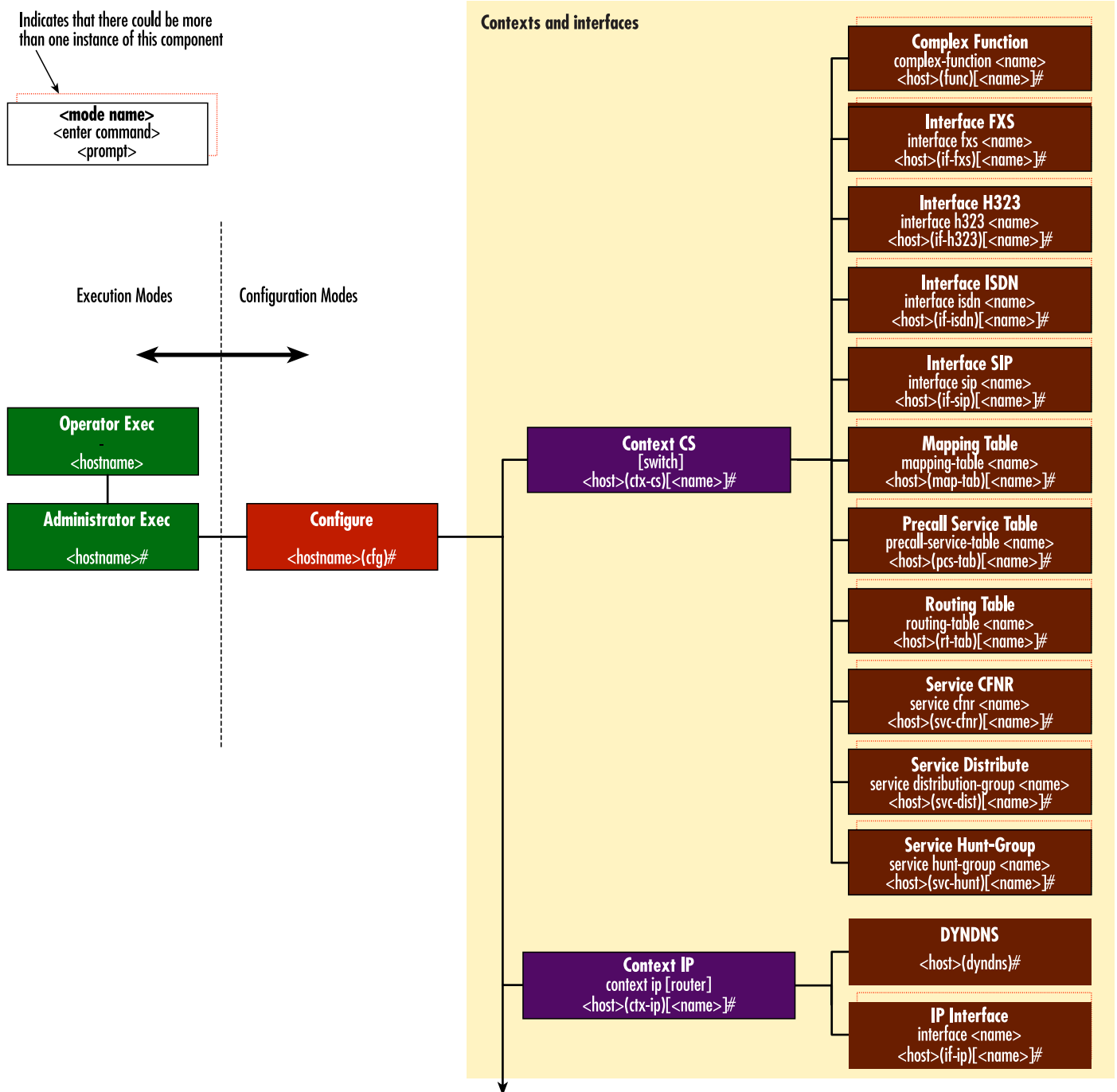


Figure 89. Mode overview, 1 of 3

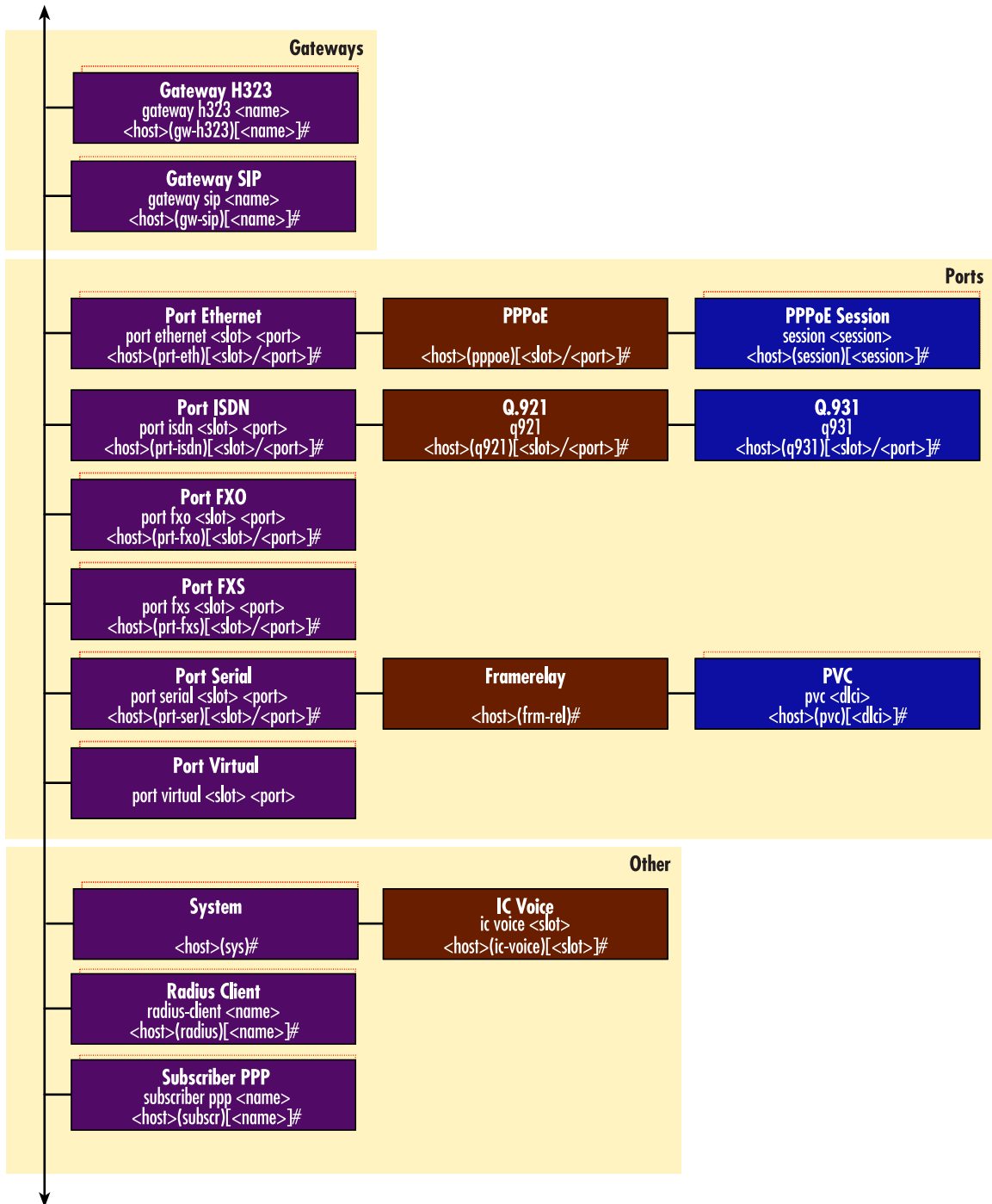


Figure 90. Mode Overview, 2 of 3

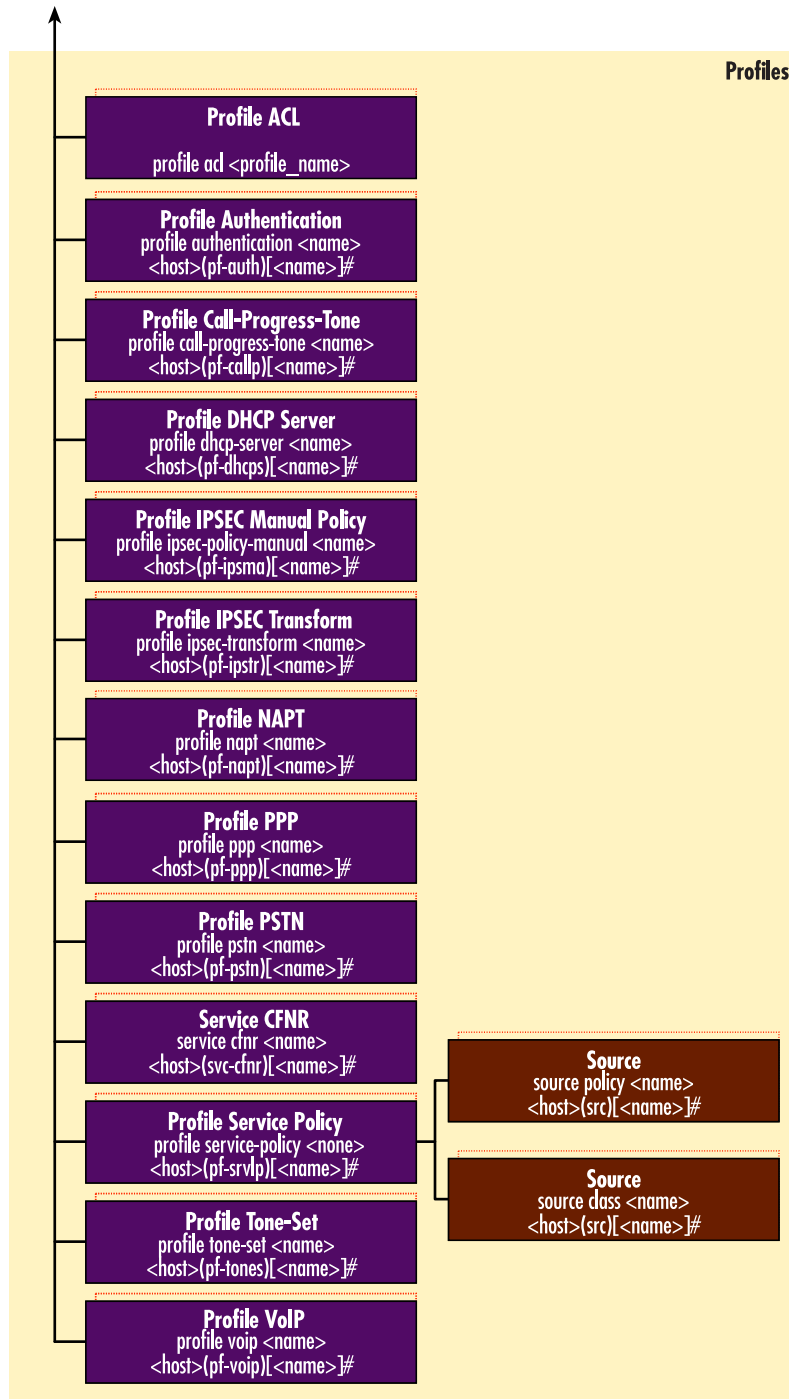


Figure 91. Mode Overview, 3 of 3

Appendix C **Command summary**

Chapter contents

Introduction	531
operator_exec	531
administrator_exec	533
configure	535
Contexts and interfaces.....	536
context_ip	536
interface	536
dyndns	537
context_cs	537
cr_table_routing	537
cr_table_mapping	537
cr_table_precall-service	537
cr_table_complex_function	538
interface_h323	538
interface_sip	538
interface_isdn	539
interface_fxs	539
interface_fxo	539
service_hunt	539
service_distribute	540
service_second-dialtone	540
Gateways	540
gateway_h323	540
gateway_sip	540
Ports.....	541
port_ethernet	541
pppoe	541
port_serial	541
framerelay	542
pvc	542
port_virtual	542
port_fxs	542
port_fxo	542
port_isdn	543
port_isdn_q921	543
port_isdn_q931	543
Profiles	543
profile_acl	543
profile_service-policy	544

source	544
profile_napt	544
profile_ppp	545
profile-ipsec-transform	545
ipsec-manual-policy	545
profile_call-progress-tone	545
profile_tone-set	545
profile_voip	546
profile_pstn	548
profile_dhcp-server	548
profile_authentication	548
Other.....	548
radius-client	548
system	548
ic_voice	548
subscriber_ppp	548
Show help	549
Show command history	549
Show RedBoot version	550
Restart system	550
Display memory content	550
Set IP addresses	550
Check network connection to remote system	551
Load a program to memory, so that it can be executed or stored in the Flash memory	551
Execute a program loaded into memory	552
Manage program images in Flash memory	552
Display images stored in Flash memory	552
Load an image into RAM so that it can be started	553
Re-initialize Flash image store	553
Create a new image in the Flash image store	554
Delete an image from the Flash image store	554
RedBoot Configuration	554
Displaying current configuration	554
Modify configuration	555
Re-initialize configuration to default values	555
Read data from EEPROM	555
Enable/Disable cache	555

Introduction

This command summary is valid for SmartWare Release 3.10. Commands in future SmartWare releases may be different. The information provided in this chapter is subject to change without notice. The command summary 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 92](#).

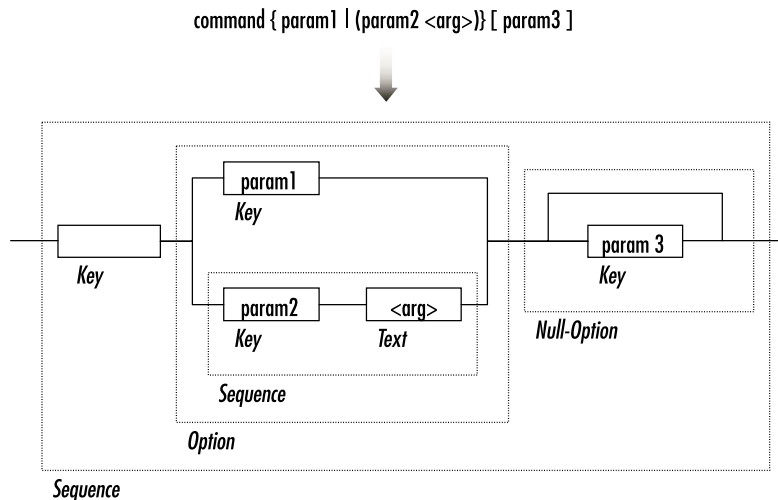


Figure 92. EBNF syntax

operator_exec

```

dropin
ping <address> [<number> ] [timeout <seconds> ] [packet-size <packet_size> ] [ttl
  <ttl> ]
dns-lookup <text_hostname>
traceroute <ip_host> [probe-count <probe_count> ] [timeout <seconds> ] [destination-
  port <port_number> ] [min-ttl <min_ttl> ] [max-ttl <max_ttl> ] [verbose ] [packet-
  size <packet_size> ] [mtu ]

```

```

[no] call [] <local-number> {(reject-all ) | (offer-state
    {trying|proceeding|alerting} [inband-info ] ) | (dial <remote-number> {(dest-
    table ) | (dest-interface ) | (dest-service ) } ) | (proceeding [inband-info no-
    inband-info ] ) | (alerting [inband-info no-inband-info ] ) | accept | suspend |
    resume | inband-info | (user-input <user-input> ) | (drop [] ) }
clear
[no] play <file> {local | (rtp <address> <port> ) }
show clock
show uptime
show ip route
show dsp {<slot> | (statistics <slot> ) | (channel statistics <slot> ) | (sw-version
    <slot> ) | (test-result <slot> ) }
show profile voip [<show_name> ]
show profile pstn []
show profile tone-set [<show_name> ]
show profile call-progress-tone [<show_name> ]
show ip interface [<interface_name> ] [router ]
show napt interface <ip_interface_name_show> [router ]
show rip [interface <ip_interface_name_show> [router ] ]
show port isdn [<print-slot> <print-port> ] [{status|calls} ] [detail <detail> ]
show port fxs (<print-slot> <print-port> )
show port fxo [<print-slot> <print-port> ] [detail <detail> ]
set port fxo [<print-slot> <print-port> ] [{onhook|offhook} ]
show port ethernet [<print-slot> <print-port> ]
show port serial [<print-slot> <print-port> ]
show framerelay [pvc <print-dlci> ]
show ppp {links|networks} [detail <level> ]
show pppoe [detail <level> ]
show log [{event|reset|boot} ]
show log login
show service-policy [interface <interface-name> [router ] ]
show version
show licenses
erase licenses: [<feature> ]
install license <license>
jobs
fg <job>
terminal width {<value> | default }
terminal height {<value> | default }
show terminal
[no] terminal more
[no] terminal {idle-time-logout|absolute-time-logout} <value>
[no] terminal mark <value> [{bold|underscore|blink|reverse} ]
logout
su <account>
who
help
show history
show version cli
show smi
show profiler-runtime
    (detail <detail> )
    (slot <print-slot> )
    (name <print-profiler> )
profiler-runtime reset [<slot> ]

```

```

show power-management [time <time> ]
eeprom read <file> [id <id> ] [from <from> to <to> ]
eeprom write <file> [id <id> ]
no

```

administrator_exec

```

enable
copy {{running-config|factory-config|startup-config|system:running-config} |
    {cli:|preferences:} | <src> | } {{running-config|startup-config|system:running-
    config|flash:|licenses:} | {cli:|preferences:} | <dest> | }
erase {{startup-config} | {cli:|preferences:} | <config> }
edit <file>
debug all
show dyndns
[no] debug dsp error
[no] debug dsp events
[no] debug dsp t30
[no] debug dsp ifp
[no] debug voip events
[no] debug voip dejitter
[no] debug dyndns
[no] debug voip rtp
[no] debug voip nte
[no] debug voip demux
[no] debug voip cs
[no] debug voip t38 events
[no] debug voip t38 dejitter
[no] debug fxs
[no] debug fxo
[no] debug isdn {(event <slot> <port> {all|layer1|layer2|layer3} ) | (error ) }
[no] debug snmp private-mib
[no] debug h235-security [detail <detail> ]
[no] debug gateway h323 [error signaling ras h245 datapath ca caerr channels cm cmapi
    cmapicb cmerr debug efrm li liinfo namechan pdlapi pdlchan pdlcomm pdlconf
    pdlencode pdlerror pdlfnerr pdlprint pdlprnerr pdlprnwrn pdlsm pdlsrc pdlmisc
    pdlmtask pdlplist pdltimer per pererr q931 ra rasctrl rasindb seli timer tpktchan
    tunnctrl udpchan unreg vt transport tcp ema memory faststart ]
[no] debug gateway sip {error | transport | registration | signaling | datapath }
    [detail <detail> ]
dsp {(up [<slot> ] ) | (down [<slot> ] ) | (reconfigure [<slot> ] ) | (test [<slot>
    ] ) | (channel {(restart [<slot> [<channel> ] ] ) | (state {(reset [<slot>
    [<channel> ] ] ) | (closed [<slot> [<channel> ] ] ) | (opened [<slot> [<channel> ]
    ] ) } ) } ) | (tone {(play <slot> <channel> <id> ) | (stop <slot> <channel> ) } )
    | (dtmf {(play <slot> <channel> <digit> ) } ) }
[no] debug cli
[no] debug debug
[no] debug acl [{in|out} [detail <detail> ] ]
[no] debug rtm
[no] debug rip
[no] debug plugin
[no] debug sntp-client
[no] debug dhcp-server
show dhcp-server
[no] debug dhcp-client

```

```

show dhcp-client
[no] debug ppp [{all|tx-hdlc|rx-hdlc|tx-control|rx-control|tx-option|rx-
  option|timer|state-machine|control|management|authentication|error} ]
[no] debug pppoe [{all|tx-discovery|rx-discovery|tx-session|rx-session|timer|state-
  machine|control|management|error} ]
[no] debug serial
[no] debug framerelay [{all|error|lmi|packet|management} ]
[no] debug ipsec
[no] debug db
[no] debug ccisdn [error signaling datapath ]
[no] debug ccfxs
[no] debug ccfxo
show accounts
show {nvram: | {running-config|factory-config|startup-config|system:running-config}
  | {cli:|preferences:} | <config> }
show crc {{running-config|factory-config|startup-config|system:running-config} |
  {cli:|preferences:} | <config> }
show profile napt [<napt-profile_name_show> ]
show subscriber ppp [<subscriber_ppp_name_show> ]
show profile ppp [<profile_ppp_name_show> ]
show ipsec security-associations
show profile ipsec-policy-manual
show h235-security
show (gateway h323 [] {config | status | stack-config | (interface [] ) | (call
  [<show_endpoint> ] ) | (vt-node <vt-node> ) } ) [detail <detail> ]
[no] h323 [] {(drop-call <exec_endpoint> ) | (change-codec <exec_endpoint>
  {g711alaw64k | g711ulaw64k | g723_6k3 | g729 | transparent | t38_udp } ) |
  (resource-tracking <stack-depth> ) | flush-resource-tracker }
show (gateway sip [] {config | status | (interface [] ) | (call [<show_endpoint> ] )
  } ) [detail <detail> ]
sip [] {(drop-call <exec_endpoint> ) | (change-codec <exec_endpoint> {g711alaw64k |
  g711ulaw64k | g723_6k3 | g729 | transparent | t38_udp } ) }
show profile ipsec-transform
show profile service-policy [<arbiter-name> ]
show datapath {context | termination } [detail <detail> ]
show call-control [] {(provider [] ) | call | status } [detail <detail> ]
call-control [] call drop {(call <call> ) | all }
show call-router [] {config|status} [] [detail <detail> ]
test call-router [] [e164 <test_value_e164> ] [type-of-number
  {unknown|international|national|network-specific|subscriber|abbreviated} ]
[numbering-plan {unknown|isdn-telephony|data|telex|national-standard|private} ]
[ip <test_value_ip> ] [uri <test_value_uri> ] [name <test_value_name> ] [itc
  {speech|unrestricted-digital|restricted-digital|3k1-audio|7k-audio|video} ] [pi
  {allowed|restricted|interworking|default} ] [si {user-not-screened|user-
  passed|user-failed|network|default} ] [called-e164 <test_value_called-e164> ]
[called-type-of-number {unknown|international|national|network-
  specific|subscriber|abbreviated} ] [called-numbering-plan {unknown|isdn-
  telephony|data|telex|national-standard|private} ] [called-ip <test_value_called-
  ip> ] [called-uri <test_value_called-uri> ] [called-name <test_value_called-name>
  ] [calling-e164 <test_value_calling-e164> ] [calling-type-of-number
  {unknown|international|national|network-specific|subscriber|abbreviated} ]
[calling-numbering-plan {unknown|isdn-telephony|data|telex|national-
  standard|private} ] [calling-ip <test_value_calling-ip> ] [calling-uri
  <test_value_calling-uri> ] [calling-name <test_value_calling-name> ] [calling-pi
  {allowed|restricted|interworking|default} ] [calling-si {user-not-screened|user-

```



```

    passed|user-failed|network|default} ] [time <test_value_time> ]
show snmp
show snmp-client
[no] debug call-control [detail <level> ]
[no] debug call-router [detail <level> ]
show profile acl [<acl_name> ]
[no] debug aaa
show profile authentication []
show radius-client []
reload
show pluginframe repository
show flashserver {general | sect
    or | file-part-list | file-list | error-file-list }
flashserver {garbage-removing }
crash {alignment|illegal-opcode|privileged-instr|trap|low-
    memory|overload|freeze|block-kernel|block-user|block-cli|block-router-mutex}
show log supervisor
mem {(stat ) | (dump [{hex|srec} <address> [<size> ] [{byte|word|long} ] ) ) | (get
    {byte|word|long} <address> ) | (set {byte|word|long} <address> <value> ) }
show memory stat
spr {(get <register> ) | (set <register> <value> ) }
show task [trace <tid> ]
rip flush-routes
ecm state <ecm> {initial|prepared|linked|started}
show ecm
show db [<table> ]
snake
show command stack
show bootloader info
show image info
show board-descriptor
show test
test {all | (<test> [<param> ] ) }
show memory-usage
show rtm
end

```

configure

```

configure
cli version <version>
terminal {console|telnet} use authentication
[no] administrator <account> password
[no] operator <account> password
[no] banner <banner>
clock set <time>
[no] {dns | dns-client } {{{domain-name-server | server } <ip_nameserver> ) |
    (cache <num_cachesize> ) }
[no] {dns-relay | (dns relay ) }
[no] webserver
    (port <port> )
    (language {en|de} )
[no] snmp community <community> {ro|rw}
[no] snmp target <ipAddress> security-name <community>
[no] snmp host <ipAddress> security-name <community>
[no] snmp-client

```

```

ntp-client server {primary|secondary} <server_address>
    (port <ntp_port> )
    (version <version_number> )
ntp-client operating-mode {unicast | multicast | anycast }
ntp-client anycast-address <ip_anycast_address> [port <ntp_port> ]
ntp-client local-port <ntp_port>
ntp-client poll-interval <number_pollinterval>
[no] ntp-client local-clock-offset
[no] ntp-client root-delay-compensation
ntp-client gmt-offset {+|-} <time_gmtoffset>
[no] ntp-client secure-mode
[no] ntp-client authentication
power-management low-power-mode {off|doze|dozeVx}
system hostname <string>
system location <string>
system contact <string>
system supplier <string>
system provider <string>
system subscriber <string>

```

Contexts and interfaces

context_ip

```

context ip [router ]
[no] dhcp-server
dhcp-server clear-lease {all | <address> }
dyndns reset
[no] dhcp-server use <name>
[no] route <destaddr> <destmask> {<gwaddr> | <interface> } [<metric> ]

```

interface

```

[no] interface <ip_interface_name>
ipaddress {unnumbered | dhcp | (<ip_address> <ip_mask> ) }
mtu <mtu>
[no] point-to-point
[no] icmp router-discovery
[no] icmp redirect accept
[no] icmp redirect send
[no] use profile acl <acl_profile_name> {in|out}
dhcp-client {renew|release}
[no] {cos | traffic-class } <cos_group>
[no] use profile service-policy <arbiter-name> {in | out }
[no] rip listen
rip receive version {1|2|1or2}
rip send version {1|2|1compatible}
[no] rip split-horizon
[no] rip supply
[no] rip announce static
[no] rip announce host
[no] rip announce default
[no] rip announce self-as-default
[no] rip learn default
[no] rip learn host
[no] rip auto-summary

```

```

[no] rip poison-reverse
[no] rip route-holddown
rip default-route-value <default_route_value>
[no] use profile napt <napt-profile_name>
[no] tcp adjust-mss {rx|tx} {mtu | <mss> }
exit

```

dyndns

```

dyndns
[no] hostname <host> [wildcard ]
[no] authentication <user>
[no] service {custom | dynamic | static }
[no] observe <ifname>
[no] mail-exchanger <host> [backup-mx ]
exit
exit

```

context_cs

```

[no] context cs []
[no] digit-collection timeout <timeout>
[no] digit-collection terminating-char <char>
[no] address-completion timeout <timeout>
[no] national-prefix <prefix>
[no] international-prefix <prefix>
[no] shutdown

```

cr_table_routing

```

[no] routing-table {({called-e164|called-type-of-number|called-numbering-
  plan|called-ip|called-uri|called-name|calling-e164|calling-type-of-
  number|calling-numbering-plan|calling-ip|calling-uri|calling-
  name|calling-pi|calling-si|itc|day-of-week|date|time} ) | }
[no] route {({(dest-table ) | (dest-interface ) | (dest-service ) } [] ) |
  none }
exit

```

cr_table_mapping

```

[no] mapping-table {({called-e164|called-type-of-number|called-numbering-
  plan|called-ip|called-uri|called-name|calling-e164|calling-type-of-
  number|calling-numbering-plan|calling-ip|calling-uri|calling-
  name|calling-pi|calling-si|e164|type-of-number|numbering-
  plan|ip|uri|name|itc|pi|si|day-of-week|date|time} to {called-e164|called-
  type-of-number|called-numbering-plan|called-ip|called-uri|called-
  name|calling-e164|calling-type-of-number|calling-numbering-plan|calling-
  ip|calling-uri|calling-name|calling-pi|calling-si|e164|type-of-
  number|numbering-plan|itc|ip|uri|name|pi|si} ) | }
[no] map to []
exit

```

cr_table_precall-service

```

[no] precall-service-table
[no] map to
exit

```

cr_table_complex_function

```
[no] complex-function
[no] execute {( <index> ) | }
exit
```

interface_h323

```
[no] interface h323
[no] bind gateway
[no] route {call} {(dest-table ) | (dest-interface ) | (dest-service ) }
[no] use mapping-table {in|out}
[no] remote <ip-address> [<port> ]
[no] remoteip <ip-address>
[no] remoteport <port>
[no] status-inquiry [timeout <timeout> ]
[no] clip-clir-support
[no] via-address-support
[no] overlap-sending-support
itc {rx|tx} {transparent|speech|unrestricted-digital|restricted-digital|3k1-
  audio|7k-audio|video}
[no] early-disconnect
[no] use profile voip <profile>
[no] use profile tone-set <profile>
exit
```

interface_sip

```
[no] interface sip
[no] bind gateway
[no] route {call} {(dest-table ) | (dest-interface ) | (dest-service ) }
[no] use mapping-table {in|out}
[no] remote <host> [<port> ]
[no] early-connect
[no] early-disconnect
[no] remote-party-id
[no] phone-context <name>
address-translation outgoing-call from-header
  (user-part {(fix <user> ) | call } )
  (host-part {(fix <host> [<port> ] ) | call | domain | local-ip } )
address-translation outgoing-call to-header
  (user-part {(fix <user> ) | call } )
  (host-part {(fix <host> [<port> ] ) | interface | call | domain | default-
    server } )
address-translation outgoing-call remote-party-id-header
  (user-part {(fix <user> ) | call } )
  (host-part {(fix <host> [<port> ] ) | call | domain | local-ip } )
[no] address-translation incoming-call calling-e164
  {(fix <value> ) | remote-party-id-header | from-header }
[no] address-translation incoming-call calling-uri
  {(fix <value> ) | remote-party-id-header | from-header | domain }
[no] address-translation incoming-call calling-name
  {(fix <value> ) | remote-party-id-header | from-header }
[no] address-translation incoming-call called-e164
  {(fix <value> ) | request-uri | to-header }
[no] address-translation incoming-call called-uri
  {(fix <value> ) | domain | local-ip | request-uri | to-header }
```

```
[no] address-translation incoming-call called-name
      {(fix <value> ) | to-header }
[no] use profile voip <profile>
[no] use profile tone-set <profile>
exit
```

interface_isdn

```
[no] interface isdn
[no] route {call} {(dest-table ) | (dest-interface ) | (dest-service ) }
[no] use mapping-table {in|out}
[no] dtmf-dialing
[no] use profile pstn
[no] use profile tone-set <profile>
[no] call-waiting
[no] isdn-date-time
exit
```

interface_fxs

```
[no] interface fxs
[no] route {call} {(dest-table ) | (dest-interface ) | (dest-service ) }
[no] use mapping-table {in|out}
[no] use mapping-table {precall-service}
[no] call-hold
[no] call-waiting
[no] additional-call-offering
[no] call-transfer
[no] caller-id-presentation {pre-ring|mid-ring}
[no] subscriber-number <number>
[no] use profile tone-set <profile>
[no] use profile pstn
exit
```

interface_fxo

```
[no] interface fxo
[no] route {call|precall} {(dest-table ) | (dest-interface ) | (dest-service
  ) }
[no] use mapping-table {in|out}
[no] use mapping-table {precall-service}
[no] disconnect-signal {battery-reversal|loop-break|busy-tone}
[no] connect-signal {battery-reversal|tax-pulse}
ring-number <number>
dial-after {dialtone | (timeout <seconds> ) }
[no] use profile tone-set <profile>
[no] use profile pstn
exit
```

service_hunt

```
[no] service hunt-group
[no] cyclic
[no] timeout <timeout>
[no] drop-cause
[no] route {call} [<index> ] {(dest-table ) | (dest-interface ) | (dest-
  service ) }
exit
```

service_distribute

```
[no] service distribution-group
[no] cyclic
[no] timeout <timeout>
[no] max-concurrent <max-concurrent>
[no] route {call} [<index> ] {(dest-table ) | (dest-interface ) | (dest-
  service ) }
exit
```

service_second-dialtone

```
[no] service second-dialtone
[no] route {call|announcement} {(dest-table ) | (dest-interface ) | (dest-
  service ) }
[no] use profile tone-set <profile>
exit
exit
```

Gateways**gateway_h323**

```
[no] gateway h323 []
[no] supported-prefix <prefix>
[no] alias {h323-id|e164} <alias>
[no] faststart
[no] early-h245
[no] h245-tunneling
terminal-type {terminal|gateway}
[no] ras
call-signaling-port <port>
[no] timeout {(response <timeout> ) | (connect <timeout> ) }
[no] gatekeeper-discovery {(auto [<id> ] ) | (manual <ip-address> <ras-port>
  [<id> ] ) }
h235-security version {{v1|v2} }
[no] h235-security password
h235-security time-window <time-window>
[no] h235-security general-id <general-id>
[no] h235-security sender-id <sender-id>
[no] h235-security ras-auth-int-tx
[no] h235-security ras-auth-int-rx
[no] h235-security q931-auth-int
[no] h235-security
[no] bind interface <interface> [router ]
[no] shutdown
[no] use profile voip <profile>
exit
```

gateway_sip

```
[no] gateway sip []
[no] domain <domain>
call-signaling-port <port>
[no] default-server <host>
  <port>
  [loose-router strict-router ]
```

```

[no] nat-traversal nortel
[no] registrar <host>
    <port>
    use-default-server
[no] registration-lifetime <lifetime>
[no] user <user> {
    (display-name <display-name> )
    (phone-context <phone-context> )
    | [<display-name> ] }
[no] authentication <realm> <user> [default ]
[no] session-timer [<seconds> ]
call-transfer-version {2 | 5 }
session-timer-version {4 | 8 }
[no] bind interface <interface> [router ]
[no] shutdown [graceful ]
[no] use profile voip <profile>
exit

```

Ports

port_ethernet

```

port ethernet <slot> <port>
medium {auto | ({10 | 100 } {half | full } ) }
encapsulation {ip|pppoe|multi}
frame-format {standard|dot1q}
[no] vlan [<vlan_id> ]
[no] map cos <cos> to <class>
[no] bind interface <ip_interface_name> [router ]
[no] shutdown

```

pppoe

```

pppoe
pppoe_session
    [no] session <session>
    [no] service <service>
    [no] access-concentrator <access-concentrator>
    [no] use profile ppp <profile_ppp_name>
    [no] bind {(interface <interface> [router ] ) | (subscriber
        <ppp_subscriber_name> ) }
    [no] shutdown
    exit
exit
exit

```

port_serial

```

port serial <slot> <port>
[no] encapsulation {framerelay|ppp}
hardware-port {v35 | x21 }
transmit-data-on-edge {positive | negative }
crc-type {crc16 | crc32 }
length <length>
threshold <threshold>
mask <mask>

```

```

address {address-1 | address-2 | address-3 | address-4 } <address>
[no] use profile ppp <profile>
[no] bind {(interface <ip_interface_name> [router ] ) | (subscriber
  {<ppp_subscriber_name> | (authentication {(chap pap ) | {chap|pap} } ) } ) }
[no] shutdown

```

framerelay

```

framerelay
lmi-type {ansi | gof | itu }
[no] keepalive [<keepalive> ]
[no] fragment <size>
[no] use profile service-policy <arbiter-name> {in | out }

```

pvc

```

[no] pvc <dlci>
encapsulation {rfc1490 }
[no] bind interface <ip_interface_name> [router ]
[no] fragment <size>
[no] shutdown
exit
exit
exit

```

port_virtual

```

[no] port virtual <slot> <port>
[no] tunnel <peer_port>
[no] encapsulation <encapsulation>
[no] bind <bind>
[no] shutdown
exit

```

port_fxs

```

port fxs <slot> <port>
end-of-call-signaling {busy-tone | (loop-break <key_dur> ) }
flash-hook-duration <key_dur>
[no] battery-reversal [smooth ]
caller-id format {bell|etsi}
[no] caller-id attenuation <att>
[no] caller-id start-with-line-reversal
[no] use profile fxs
[no] pulse-dialing
tax-pulse-modulation {bursts-12khz|bursts-16khz|polarity-reversal}
[no] encapsulation {cc-fxs}
[no] bind interface []
[no] shutdown
exit

```

port_fxo

```

port fxo <slot> <port>
flash-hook-duration <key_dur>
[no] use profile fxo
[no] caller-id format {bell|etsi}
[no] encapsulation {cc-fxo}

```



```
[no] bind interface []
[no] shutdown
```

port_isdn

```
port {bri|el|t1} <slot> <port>
clock {master|slave|auto}
[no] linecode {ami|hdb3|b8zs}
[no] encapsulation {q921}
[no] bind {(subscriber <binding> ) }
[no] shutdown
```

port_isdn_q921

```
q921
[no] permanent-layer2
protocol {pp|pmp}
uni-side {net|user|auto}
[no] encapsulation {q931}
[no] bind gateway <binding>
```

port_isdn_q931

```
q931
protocol {dss1 | pss1 | ni2 | ntt }
uni-side {net|user}
bchan-number-order {ascending|descending|ascending-cyclic|descending-
cyclic}
[no] signalling-rule {etsi|pss1old}
[no] max-calls <maxcalls>
[no] channel-range <low> <high>
[no] ais-blue-alarm
[no] encapsulation {cc-isdn}
[no] bind interface []
exit
exit
exit
exit
exit
exit
```

Profiles

profile_acl

```
[no] profile acl <profile_name>
{permit|deny} {( {ip|ah|esp|gre} {any | (host <src_ip> ) | (<src_ip>
<src_wildcard> ) } {any | (host <dst_ip> ) | (<dst_ip> <dst_wildcard> ) } )
| ( {tcp|udp|sctp} {any | (host <src_ip> ) | (<src_ip> <src_wildcard> ) } [eq
<src_port> lt <src_port> gt <src_port> range <src_port_from> <src_port_to> ]
{any | (host <dst_ip> ) | (<dst_ip> <dst_wildcard> ) } [eq <dst_port> lt
<dst_port> gt <dst_port> range <dst_port_from> <dst_port_to> ] ) | (icmp
{any | (host <src_ip> ) | (<src_ip> <src_wildcard> ) } {any | (host <dst_ip>
) | (<dst_ip> <dst_wildcard> ) } [name {administratively-
prohibited|alternate-address|conversion-error|dod-host-prohibited|dod-net-
prohibited|echo|echo-reply|general-parameter-problem|host-isolated|host-
precedence-unreachable|host-redirect|host-tos-redirect|host-tos-
```

```

unreachable|host-unknown|host-unreachable|information-reply|information-
request|mask-reply|mask-request|mobile-redirect|net-redirect|net-tos-
redirect|net-tos-unreachable|net-unreachable|network-unknown|no-room-for-
option|option-missing|packet-too-big|parameter-problem|port-
unreachable|precedence-unreachable|protocol-unreachable|reassembly-
timeout|redirect|router-advertisement|router-solicitation|source-
quench|source-route-failed|time-exceeded|timestamp-reply|timestamp-
request|traceroute|ttl-exceeded|unreachable} type <type> type <type> code
<code> ] ) } [[tos {{max-reliability|max-throughput|min-delay|min-monetary-
cost|normal} | <tos-value> } ] [precedence {{critical|flash|flash-
override|immediate|internet|network|priority|routine} | <precedence> } ]
dscp <dscp> [mask {1|3|7|15|31} ] ] [{cos | traffic-class } <cos> {cos-rtcp |
rtcp-traffic-class } <cos_rtcp> <cos_rtcp> ] [ipsec-policy <ipsec_policy> ]
exit

```

profile_service-policy

```

[no] profile service-policy <arbiter-name>
mode {shaper | burst-shaper | wfq | burst-wfq }
[no] rate-limit <value>
    (header-length <option-value> )
    atm-modem
    (voice-margin <voice-margin-value> )
[no] queue-limit <value>
[no] set ip dscp <value>
[no] set ip precedence <value>
[no] set ip tos <value>
[no] set layer2 cos <value>
[no] debug queue statistics [detail <value> ]

```

source

```

[no] source {{{traffic-class | class } <source-name> ) | (policy <source-
name> ) }
[no] rate {<value> | remaining }
[no] share <value>
[no] random-detect [<value> ]
[no] priority
[no] police <value> burst-size <option-value>
[no] queue-limit <value>
[no] set ip dscp <value>
[no] set ip precedence <value>
[no] set ip tos <value>
[no] set layer2 cos <value>
[no] debug queue statistics [detail <value> ]
exit
exit

```

profile_napt

```

[no] profile napt <napt-profile_name>
[no] range <local_ip1> <local_ip2> <global_ip> [<global_ip2> ]
[no] static <local_ip1> <global_ip>
[no] static {udp|tcp} <local_ip1> <local_port> [<global_ip> ] [<global_port> ]
[no] static {udp|tcp} <local_port> <local_ip1>
exit

```

profile_ppp

```
[no] profile ppp <profile_ppp_name>
lcp-configure-request interval <interval> max <max>
lcp-configure-nak max <max>
lcp-terminate-request interval <interval> max <max>
lcp-echo-request interval <interval> max <max>
mtu min <min> max <max> [ignore-link ]
mru min <min> max <max> [ignore-link ]
accm <value>
[no] authentication {(chap pap ) | {chap|pap} } interval <interval> max <max>
[no] callback {active|passive|both} interval <interval> max <max>
ipcp-configure-request interval <interval> max <max>
ipcp-configure-nak max <max>
ipcp-terminate-request interval <interval> max <max>
[no] van-jacobson {compression|decompression} max-slots <max>
[no] local-address-autoconfig
exit
```

profile-ipsec-transform

```
[no] profile ipsec-transform
[no] esp-encryption {(aes-cbc [128 192 256 ] ) | (des-cbc [64 ] ) | (3des-cbc
  [128 192 ] ) | (null ) }
[no] esp-authentication {hmac-md5-96 | hmac-shal-96 }
[no] ah-authentication {hmac-md5-96 | hmac-shal-96 }
exit
```

ipsec-manual-policy

```
[no] profile ipsec-policy-manual
[no] use profile ipsec-transform
[no] session-key {inbound|outbound} {ah-authentication|esp-authentication|esp-
  encryption} <key>
[no] spi {inbound|outbound} {ah|esp} <spi>
[no] peer <peer>
[no] mode {tunnel|transport}
exit
```

profile_call-progress-tone

```
[no] profile call-progress-tone <name>
[no] play <duration> <freq1> <level1> [<freq2> <level2> ]
flush-play-list
high-frequency <high_frequency>
low-frequency <low_frequency>
high-frequency-level {mute | <high_frequency_level> }
low-frequency-level {mute | <low_frequency_level> }
on1 <on1>
off1 <off1>
on2 <on2>
off2 <off2>
exit
```

profile_tone-set

```
[no] profile tone-set <name>
dtmf-duration <dtmf_duration>
```

```

dtmf-interspace <dtmf_interspace>
dtmf-signal-level {mute | <dtmf_signal_level> }
[no] map {{(call_progress_tone | call-progress-tone } <internal_tone_name>
        <call_progress_tone_name> ) }
exit

```

profile_voip

```

[no] profile voip <name>
[no] codec [<index> ] {(transparent
    (rx-length <rx-length> )
    (tx-length <tx-length> )
    ) | (g711alaw64k
    (rx-length <rx-length> )
    (tx-length <tx-length> )
    (silence-suppression [voice-update-frames no-voice-update-frames ] )
    no-silence-suppression
    ) | (g711ulaw64k
    (rx-length <rx-length> )
    (tx-length <tx-length> )
    (silence-suppression [voice-update-frames no-voice-update-frames ] )
    no-silence-suppression
    ) | (g723-5k3
    (rx-length <rx-length> )
    (tx-length <tx-length> )
    (silence-suppression [voice-update-frames no-voice-update-frames ] )
    no-silence-suppression
    ) | (g723-6k3
    (rx-length <rx-length> )
    (tx-length <tx-length> )
    (silence-suppression [voice-update-frames no-voice-update-frames ] )
    no-silence-suppression
    ) | (g726-16k
    (rx-length <rx-length> )
    (tx-length <tx-length> )
    silence-suppression
    no-silence-suppression
    ) | (g726-24k
    (rx-length <rx-length> )
    (tx-length <tx-length> )
    silence-suppression
    no-silence-suppression
    ) | (g726-32k
    (rx-length <rx-length> )
    (tx-length <tx-length> )
    silence-suppression
    no-silence-suppression
    ) | (g726-40k
    (rx-length <rx-length> )
    (tx-length <tx-length> )
    silence-suppression
    no-silence-suppression
    ) | (g727-16k
    (rx-length <rx-length> )
    (tx-length <tx-length> )
    silence-suppression

```

```

no-silence-suppression
) | (g727-24k
(rx-length <rx-length> )
(tx-length <tx-length> )
silence-suppression
no-silence-suppression
) | (g727-32k
(rx-length <rx-length> )
(tx-length <tx-length> )
silence-suppression
no-silence-suppression
) | (netcoder-6k4
(rx-length <rx-length> )
(tx-length <tx-length> )
silence-suppression
no-silence-suppression
) | (netcoder-9k6
(rx-length <rx-length> )
(tx-length <tx-length> )
silence-suppression
no-silence-suppression
) | (g729
(rx-length <rx-length> )
(tx-length <tx-length> )
silence-suppression
no-silence-suppression
) }
[no] high-pass-filter
[no] post-filter
[no] dtmf-relay
[no] rtp payload-type nte <value>
[no] dtmf-mute-encoder
[no] silence-suppression
dejitter-mode {adaptive|static|static-data}
dejitter-max-delay <dejitter_max_delay>
dejitter-max-packet-loss <dejitter_max_packet_loss>
dejitter-shrink-speed <dejitter_shrink_speed>
dejitter-grow-step <dejitter_grow_step>
dejitter-grow-attenuation <dejitter_grow_attenuation>
[no] fax transmission [<index> ] {(relay {t38-udp} ) | (bypass
    {g711alaw64k|g711ulaw64k} ) }
fax redundancy {ls | low-speed } <ls_red> {hs | high-speed } <hs_red>
fax volume {-18.5|-17.5|-16.5|-15.5|-14.5|-13.5|-12.5|-11.5|-10.5|-9.5|-8.5|-
    7.5|-6.5|-5.5|-4.5|-3.5}
fax dejitter-max-delay <buffer_size>
[no] fax error-correction
[no] fax hdlc
fax max-bit-rate {2400|4800|7200|9600|12000|14400}
fax detection {ced-tone|fax-frames}
[no] modem transmission [<index> ] (bypass {g711alaw64k|g711ulaw64k} )
modem dejitter-max-delay <buffer_size>
exit

```

profile_pstn

```
[no] profile pstn
echo-canceller-hybrid-loss {0|3|6|9}
[no] echo-canceller-nlp {adaptive|silence}
[no] echo-canceller
[no] output-gain <output_gain>
exit
```

profile_dhcp-server

```
[no] profile dhcp-server <dhcps_profile_name>
network <address> <mask>
[no] include <from> <to>
lease {( <time> {{days|hours|minutes} } ) | infinite }
[no] default-router <address>
[no] domain-name <name>
[no] domain-name-server <address>
[no] netbios-name-server <address>
[no] netbios-node-type {{b-node|p-node|m-node|h-node} }
[no] bootfile <bootfile>
[no] next-server <address>
exit
```

profile_authentication

```
[no] profile authentication
[no] method [<index> ] {(radius ) | local | none }
server-timeout <timeout>
exit
```

Other

radius-client

```
[no] radius-client
radius-server <hostname>
[no] shared-secret {authentication}
exit
```

system

```
system
[no] bypass-mode
clock-source {internal | ( <slot> <port> ) }
[no] hairpinning
```

ic_voice

```
ic voice <slot>
  pcm {(law-select {aLaw | uLaw } ) }
  low-bitrate-codec {g723 | g729 }
  exit
exit
```

subscriber_ppp

```
[no] subscriber ppp <subscriber_ppp_name>
dial {in|out}
```

```

[no] authentication {(chap pap ) | {chap|pap} }
[no] identification {outbound|inbound} <id> [password ]
[no] timeout {absolute|idle} <timeout>
[no] max-sessions <max-sessions>
[no] ipaddress <address>
[no] callback [mandatory ] [destination {any|some|fix} ]
[no] callback dial-string <dial-string>
[no] bind interface <interface> [router ]
exit

```

Show help

	Command	Purpose
Step 1	help [<i>topic</i>]	Shows command help.

Show command history

	Command	Purpose
Step 1	history	Shows command history.

Use CTRL-N and CTRL-P to browse. The cursor keys (up, down) are not working.

Show RedBoot version

	Command	Purpose
Step 1	version	Shows RedBoot version.

Restart system

	Command	Purpose
Step 1	reset	Restarts the system.

Display memory content

	Command	Purpose
Step 1	dump -b <i>location</i> [-l <i>length</i>] [-s] [-1 2 4] x -b <i>location</i> [-l <i>length</i>] [-s] [-1 2 4]	Displays memory. -b: Start of memory (default: next address). -l: Length in bytes (default: 256). -s: Show as S-Record. -1: Show as bytes (8 bit). -2: Show as words (16 bit). -3: Show as longs (32 bit).

Set IP addresses

	Command	Purpose
Step 1	ip_address [-l <i>local_ip_address</i> [/ <i>mask_len</i>]] [-h <i>server</i>] [-g <i>gateway</i>]	Sets the IP address of Ethernet interface 0/0 -l: Local IP address and subnet mask. The subnet mask is specified as the number of one bits in the mask (i.e. /24 for 255.255.255.0) -h: Server IP address. Used for TFTP download. -g: Default gateway IP address.

Check network connection to remote system

	Command	Purpose
Step 1	ping [-v] [-n <i>count</i>] [-l <i>length</i>] [-t <i>timeout</i>] [-r <i>rate</i>] [-i <i>IP_address</i>] -h <i>IP_address</i>	Check network connection to a remote system, by sending ICMP echo requests and listening for ICMP echo replies. -v: Verbose mode, shows extra information. -n: Number of pings to send (Default: 10) -l: Length of request to send in bytes (Default: 64). -t: Time to wait for a response (Default: 1000mS). -r: Rate at which requests are sent in milliseconds. For example, 20 = 50 requests per seconds (Default: 1000mS) -i: The IP address RedBoot should use (Default: value set by ip_address) -h: Host to ping.

Load a program to memory, so that it can be executed or stored in the Flash memory

	Command	Purpose
Step 1	load [-r] [-v] [-h <i>host</i>] [-m <i>various</i>] [-c <i>channel</i>] [-b <i>base_address</i>] <i>file_name</i>	Loads a program via TFTP, X- or Y-modem (serial port) to memory. -r: Selects raw or ELF data mode. Use -r for firm-ware images. -v: Display a spinner to show progress. -h: IP address of host to get file from (Default: value set by ip_address) -m: Transfer mode. One of: tftp, xmodem, ymodem. -c: Channel to use for serial transfer (Default: 0). -b: Address in memory to load program. <i>file_name</i> : Name of program to load. Used for TFTP transfer.

Execute a program loaded into memory

	Command	Purpose
Step 1	go [-w <i>timeout</i>] [-i] [-s <i>script-name</i>] [<i>entry</i>]	Executes a program that has formerly been loaded to memory. The command will check whether a firmware application has been loaded to the specified address. It will abort with an error message if not. -w: Timeout before program is started. You can abort the countdown by pressing CTRL-C. -i: Start program with caches disabled. -s: Name of startup-config the program shall execute. entry: Address of program in memory.

Manage program images in Flash memory

The following commands manage program images that are stored in the Flash memory.

Display images stored in Flash memory

	Command	Purpose
Step 1	fis list [-l]	Displays images stored in Flash memory. -l: Long format, displays additional information

```
RedBoot> fis list
Id Address      Length   State      Description
-----
1  0x60030000  1693438  valid      Firmware R2.10 BUILD28015
```

```
RedBoot> fis list -l
Id Address      Length   State      Description
Entry          Load Addr Version
-----
1  0x60030000  1693438  valid      Firmware R2.10 BUILD28015
   0x01800100  0x01800100  V2.10
```

Load an image into RAM so that it can be started

	Command	Purpose
Step 1	fis load [-b <i>memory_load_address</i>] [-n <i>index</i>]	Loads an image into RAM. -b: Memory address to load image to. If this parameter is omitted, the loading address will be taken from the image file. -n: Index of image to load (Default: 1).

Re-initialize Flash image store

Note *All images are removed during this process.*

	Command	Purpose
Step 1	fis init	Re-initializes Flash image store. All memory is erased and all images are removed. The command asks for confirmation, before it starts erasing.

Create a new image in the Flash image store

	Command	Purpose
Step 1	fis create -b <i>mem_base</i> -l <i>image_length</i> [-n <i>index</i>]	Creates a new image in the Flash image store. The program image must already be present in memory. -b: Address in memory where program image is located. -l: Length of program image. Both parameters can be omitted if an image has been loaded via TFTP. -n: Index of new image to create. If not specified a new image is created in the image store. You can use this parameter to replace an existing image. This is only possible if the new image fits in the same memory region.

Delete an image from the Flash image store

	Command	Purpose
Step 1	fis delete -n <i>index</i>	Deletes an image from the Flash image store. -n: Index of image to delete. If omitted the last image is deleted. Otherwise the specified image and all following images are deleted.

RedBoot Configuration

The following commands are used to manage the RedBoot configuration. The configuration is stored in persistent memory and remains valid across system reboots.

Some of the options are changed by firmware to reflect the current firmware settings. In other words, it will set the local IP address to the address actually configured in firmware.

Displaying current configuration

	Command	Purpose
Step 1	fconfig -l	Displays current configuration.

```

RedBoot> fconfig -l
Run script at boot           : true
Boot script                  :
.. fis load
.. go

Boot script timeout (1000ms resolution) : 5
Use BOOTP for network configuration  : false
Gateway IP address           : 0.0.0.0
Local IP address             : 172.31.14.151
Local IP address mask       : 255.255.255.0

```

```

Default server IP address      : 0.0.0.0
Force console for special debug messages : false
Network debug at boot time    : false

```

Modify configuration

	Command	Purpose
Step 1	fconfig	Modifies configuration. Each option is printed to the console and you are asked to modify it. Press enter to leave the value unchanged, or use the backspace key to delete the current value and specify the new value. Note: It is not possible to retain to boot script. It has to be retyped each time you run this command. Entering an empty line ends the boot script.

Re-initialize configuration to default values

	Command	Purpose
Step 1	fconfig -i	The command asks for confirmation, before it resets all values to their defaults.

Read data from EEPROM

	Command	Purpose
Step 1	eeeprom read -e <i>eeeprom_addr</i> -b <i>mem_base</i> -l <i>length</i> [-i <i>device</i>]	Reads data from EEPROM and stores it in RAM. -e: EEPROM address to read. -b: Memory address to write data to. -l: Number of bytes to read -i: Device ID (default: 100 = Mainboard)

Enable/Disable cache

	Command	Purpose
Step 1	cache [on off]	Enables/Disables cache memory. If called without arguments, shows current cache state.

Appendix D **Internetworking terms & acronyms**

Chapter contents

Abbreviations.....	558
--------------------	-----

Abbreviations

Abbreviation	Meaning
Numeric	
10BaseT	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

Abbreviation	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
SmW	SmartWare
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	
H.323	ITU-T Voice over IP Standard
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
SN	SmartNode
ISDN	Integrated Services Digital Network
ISDN NT	ISDN Network Termination
ISDN S	ISDN S(ubscriber 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

Abbreviation	Meaning
M	
MGCP	Media Gateway Control Protocol
MIB II	Management Information Base II
Modem	Modulator – Demodulator
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

Abbreviation	Meaning
RTP	Real-time Protocol
S	
S1	SN-connection for Trunk Line
S2	SN-connection for Subscriber Line
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

Appendix E **Used IP ports & available voice codecs in the SmartWare**

Chapter contents

Used IP ports in the SmartWare	564
Available voice codecs in the SmartWare	565

Used IP ports in the SmartWare

Component	Port	Description
H.323	UDP 1719	RAS for gatekeeper connection
	TCP 1720	Call signaling port for H.323 (adjustable)
	UDP 4864...5118 (even numbers)	Voice data (RTP)
	UDP 4865...5119 (odd numbers)	Voice statistics (RTCP)
NAPT	TCP 8000-15999	NAPT port range
Telnet	TCP 23	TCP server port
Webserver	TCP 80	TCP server port
DHCP	UDP 67	Source port DHCP Server
	UDP 68	Source port DHCP Client
TFTP	UDP 69	Control port of the TFTP Server (accessed by the TFTP Client in the SmartNode)

Available voice codecs in the SmartWare

Protocol	Codec	Net Bandwidth per Call (kbps)	Min. Compression Delay (ms)	Used Bandwidth per Call (kbps, incl. IP header)	Usage
ISoIP	G.711 A-Law	64	10	96	Uncompressed, best voice quality, European audio-digitizing
	G.711 u-Law	64	10	96	Uncompressed, best voice quality, American audio-digitizing
	G.726	16, 24, 32, 40	20	32, 40, 48, 56	The G.726 is an ADPCM based codec, with small memory footprint but fairly high CPU time requirements.
	G.727	16, 24, 32	20	32, 40, 48	Embedded ADPCM. See also G.726
	G.723.1	5.3, 6.3	30	16, 17	Good voice quality at lowest bandwidth, like analog phone, acceptable delay
	G.729/ G.729a	8	10	40	Best relationship between voice quality and used bandwidth, low delay
	Netcoder	6.4, 9.6	20	22.4, 25.6	License free low bandwidth codec comparable to G.723
	Transparent	64	10	96	Transparent ISDN data, no echo cancellation
H.323	G.711 A-law	64	10	96	Uncompressed, best voice quality, European audio-digitizing
	G.711 U-law	64	10	96	Uncompressed, best voice quality, American audio-digitizing
	G.723.1	6.3	30	17	Good voice quality at lowest bandwidth, like analog phone, acceptable delay
	G.729/ G.729a	8	10	40	Best relationship between voice quality and used bandwidth, low delay
	Transparent	64	10	96	Transparent ISDN data, no echo cancellation

