PATTON Electronics Co.

SmartNode Series

SmartWare Release 2.20

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: **132111U Rev. B**
Part Number: **07MDSW_SCG_2_20**
Revised: **December 1, 2003**

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

Contents

About this guide

The objective of this *SmartWare Command Configuration Guide* is to provide information concerning the syntax and usage of the command set. The aim is to enable you to be able to consult a more detailed command description than is given in the *Software Configuration Guide*. 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 18)

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 consequently 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 407 contains the terms and their definitions that are used throughout this *SmartWare Software Configuration Guide*
- Chapter 1, "System overview" on page 21 provides an overview of the main elements of a SmartNode system.
- Chapter 9, "IP context overview" on page 101
- Chapter 23, "CS context overview" on page 275

Structure

This guide contains the following chapters and appendices:

- Chapter 1, "System overview" on page 21 provides an overview of the main elements of a SmartNode system.
- Chapter 2, "Configuration concepts" on page 29 introduces basic SmartWare configuration concepts.

- Chapter 3, "Command line interface (CLI)" on page 35 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 43 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 53 explains how to establish network-based connections to and from your SmartNode using IP interfaces and Ethernet ports.
- Chapter 6, "System image handling" on page 61 describes how to load and maintain system images and driver software.
- Chapter 7, "Configuration file handling" on page 77 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 91 describes parameters that report basic system information to the operator or administrator, and their configuration.
- Chapter 9, "IP context overview" on page 101 outlines the SmartWare *Internet protocol (IP) context*, together with its related components.
- Chapter 10, "IP interface configuration" on page 109 provides a general overview of SmartNode interfaces and describes the tasks involved in configuring them.
- Chapter 11, "NAT/NAPT configuration" on page 119 provides a general overview of Network Address Port Translation and describes the tasks involved in configuring it.
- Chapter 12, "Ethernet port configuration" on page 127 provides an overview of Ethernet ports and describes the tasks involved in configuring Ethernet ports through the Patton SmartWare.
- Chapter 13, "Link scheduler configuration" on page 137 describes how to use and configure the SmartWare, Quality of Service (QoS) features.
- Chapter 14, "Serial port configuration" on page 159 provides an overview of the serial port and describes the tasks involved in configuring the serial port through the Patton SmartWare.
- Chapter 15, "Basic IP routing configuration" on page 177 provides an overview of IP routing and describes the tasks involved in configuring static IP routing in SmartWare.
- Chapter 16, "RIP configuration" on page 183 provides an overview of the Routing Information Protocol (RIP) and describes the tasks involved in configuring RIP features within SmartWare.
- Chapter 17, "Access control list configuration" on page 193 provides an overview of IP Access Control Lists and describes the tasks involved in configuring them through the Patton SmartWare.
- Chapter 18, "SNMP Configuration" on page 207 provides overview information about Simple Network Management Protocol (SNMP) and describes the tasks used to configure those of its features supported by the Patton SmartWare.
- Chapter 19, "SNTP client configuration" on page 223 describes how to configure Simple Network Time Protocol (SNTP) client.
- Chapter 20, "DHCP configuration" on page 235 provides an overview of the Dynamic Host Configuration Control Protocol (DHCP) and describes the tasks involved in configuring them.

- Chapter 21, “PPP configuration” on page 245 describes how to configure the Point-to-Point Protocol over different link layers.
- Chapter 22, "VPN configuration" on page 263 describes how to configure the Point-to-Point Protocol over different link layers.
- Chapter 23, “CS context overview” on page 275 gives an overview of the SmartWare circuit-switching (CS) context and its associated components and describes the tasks involved in its configuration.
- Chapter 24, “CS interface configuration” on page 295 provides an overview of interfaces in the CS context and describes the tasks involved in configuring them.
- Chapter 25, “Session router configuration” on page 311 provides an overview of Session Router tables and number manipulation functions and describes the tasks involved in configuring the Session Router in SmartWare.
- Chapter 26, “Tone configuration” on page 331 gives an overview of SmartWare call-progress-tone profiles and tone-set profiles and describes the tasks involved in their configuration.
- Chapter 27, “ISDN port configuration” on page 341 provides an overview of SmartNode ISDN ports and describes the tasks involved in configuring ISDN ports in SmartWare.
- Chapter 28, “POTS port configuration” on page 353 provides an overview of SmartNode POTS ports and describes the tasks involved in configuring POTS ports in SmartWare.
- Chapter 29, “Gateway configuration” on page 359 provides an overview of ISoIP and H.323 gateways and describes the tasks involved in configuring them.
- Chapter 30, “VoIP profile configuration” on page 377 gives an overview of SmartWare VoIP profiles, how they are used and describes the tasks involved in VoIP profile configuration.
- Chapter 31, “VoIP debugging” on page 395 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 407 contains the terms and their definitions that are used throughout this *SmartWare Software Configuration Guide*.
- Appendix B, “Configuration mode overview” on page 415 illustrates the configuration modes hierarchy.
- Appendix C, “Internetworking terms & acronyms” on page 439 contains terms and definitions relating to internetworking.
- Appendix D, “Used IP ports & available voice codecs in SmartWare” on page 445.

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.
Futura bold-italic type	Parts of commands, which are related to elements already named by the user, are in ***boldface italic*** font.
Italicized Futura type	Variables for which you supply values are in *italic* 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 SmartNode
#	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

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

- SmartNode hardware platforms (see page 23)
- SmartWare embedded software (see page 24)
- SmartView Management Center tools (see page 25)

A complete SmartNode system or network, as installed in any of the application scenarios introduced in section "Applications" on page 25, is typically composed of three 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.

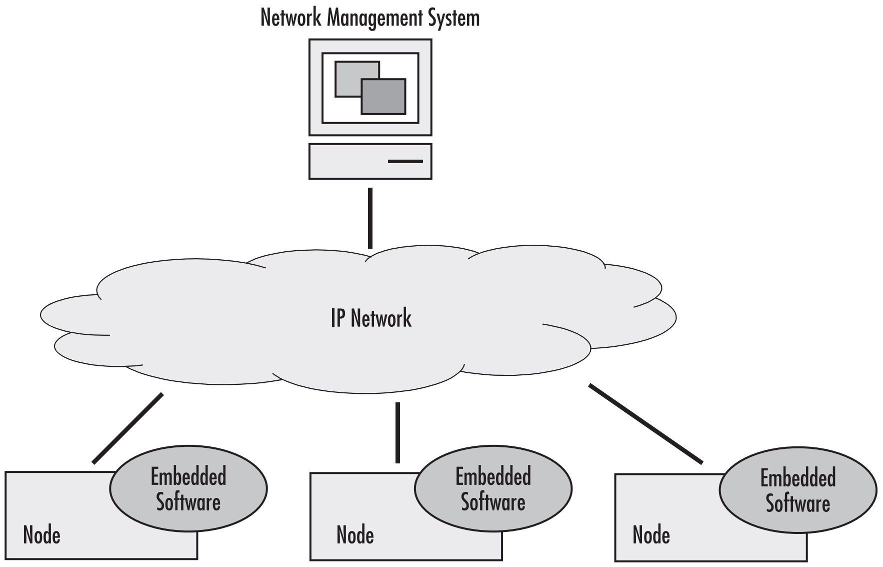


Figure 1. System overview with a SmartView Management Center

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

The basic system model of a Patton SmartNode is depicted in figure 2. 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 Voice over IP (VoIP) is supported via Patton's patented ISDN over IP (ISoIP) protocol.
- 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.

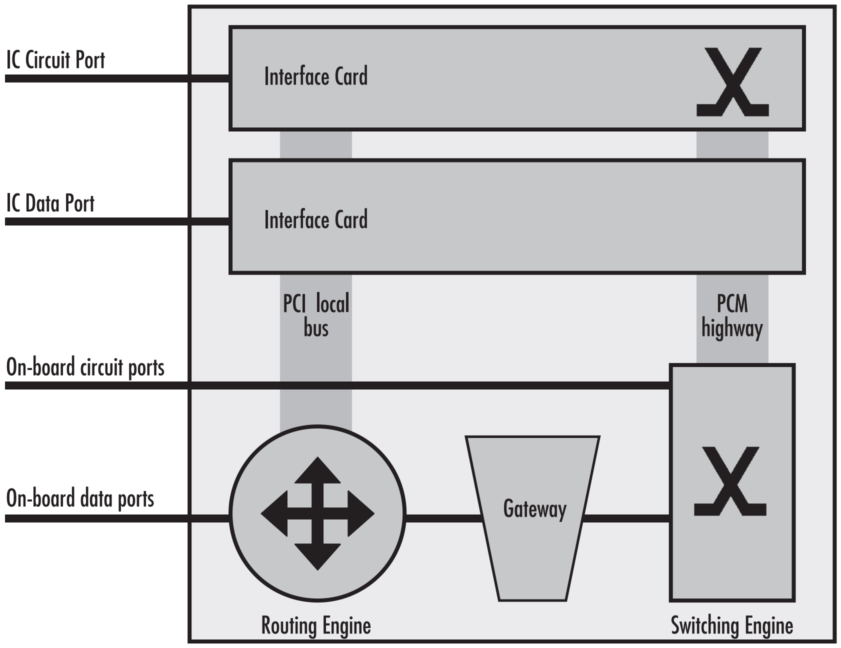


Figure 2. SmartNode System Model

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 the 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 the SmartWare release notes for build numbers and build-specific enhancements and limitations.

A SmartWare build is a binary image file. A SmartWare build is usually divided into several checksum-protected files to improve download efficiency and security. The download to the SmartNode is handled in sequence using a download *batchfile*. Refer to chapter 6, "System image handling" on page 61 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 the SmartWare application. The boot loader 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 boot loader is installed in the factory and requires no upgrading.
- The *PMC loader* initializes PMC interface cards when mounted in SmartNode 2000 series devices. It checks hardware versions and determines whether compatible PMC drivers are available. The PMC loader may be upgraded together with a SmartWare release.
- *PMC driver* software performs the runtime tasks on PMC interface cards mounted in SmartNode 2000 series devices. The PMC drivers are interface card specific and also have build numbers. Refer to the Smart-

Ware 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 flash memory file system.

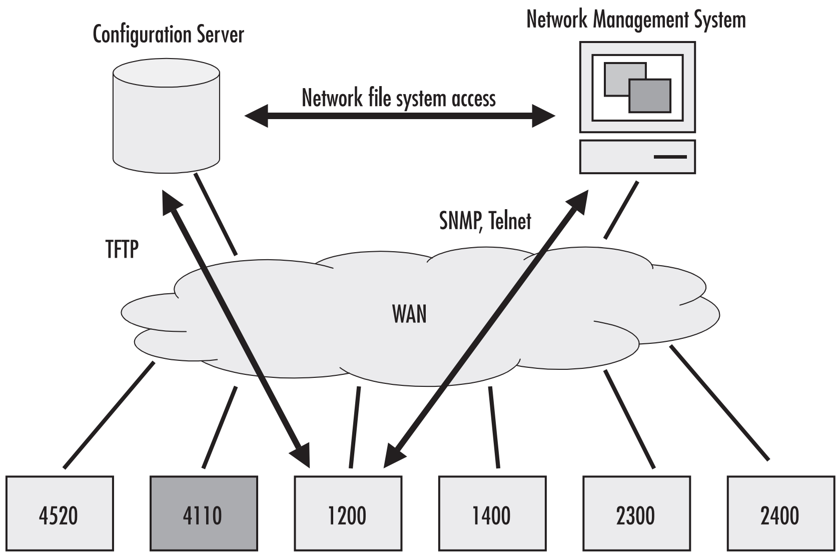


Figure 3. SmartNode Management System

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.

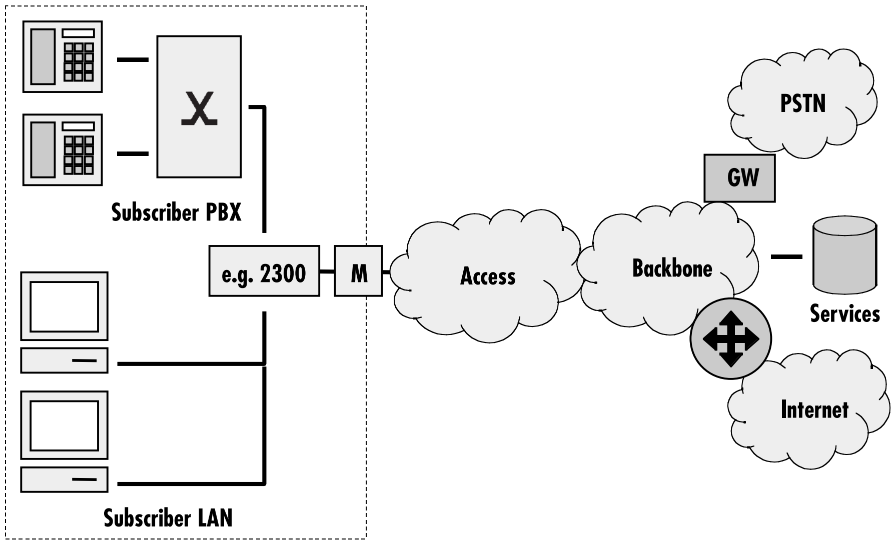


Figure 4. Typical carrier network application with a SmartNode 2300

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 on an external modem (M) the SmartNode can connect to leased lines or any bridged-Ethernet broadband access.

Enterprise networks

In company owned and operated wide are 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.

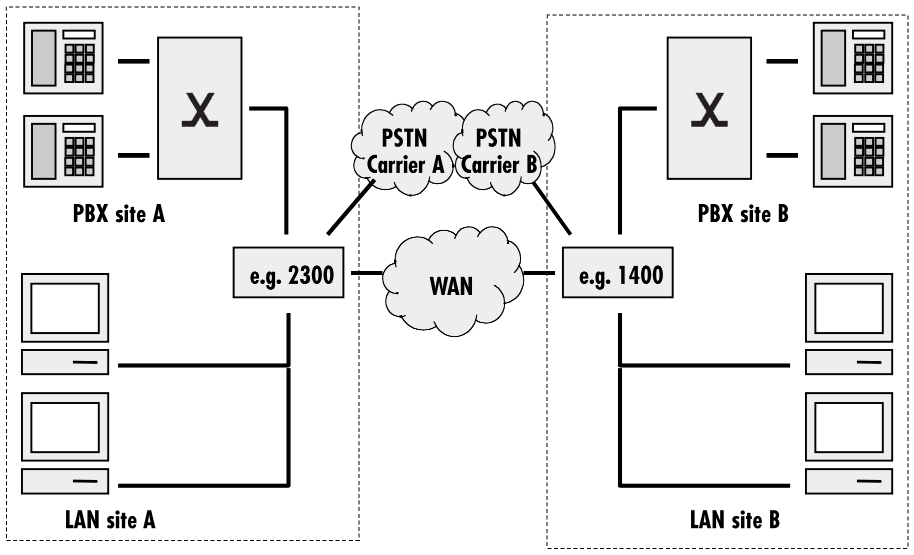


Figure 5. Typical Enterprise network with a SmartNode 1400 and 2300

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.

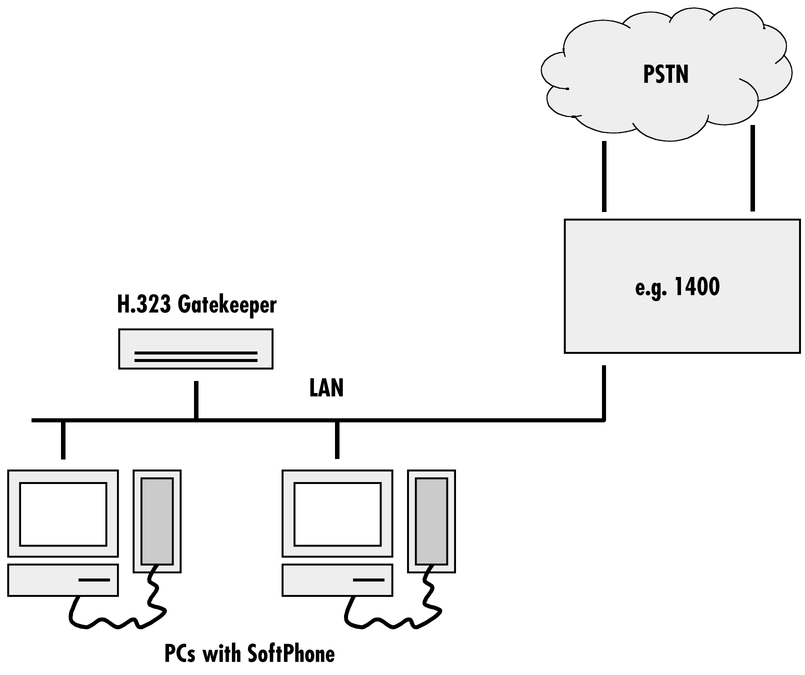


Figure 6. Typical LAN telephony system with a SmartNode 1400 gateway

LAN telephony

With its Voice-over-IP gateway features the SmartNode can be used as a standalone gateway for H.323 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.

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

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.

Even if you do not like to read manuals and user guides, nevertheless we strongly recommend 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.

The chapter includes the following sections:

- Contexts and gateways (see page 32)
- Interfaces, ports, and bindings (see page 32)
- Profiles and Use commands (see page 34)

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.

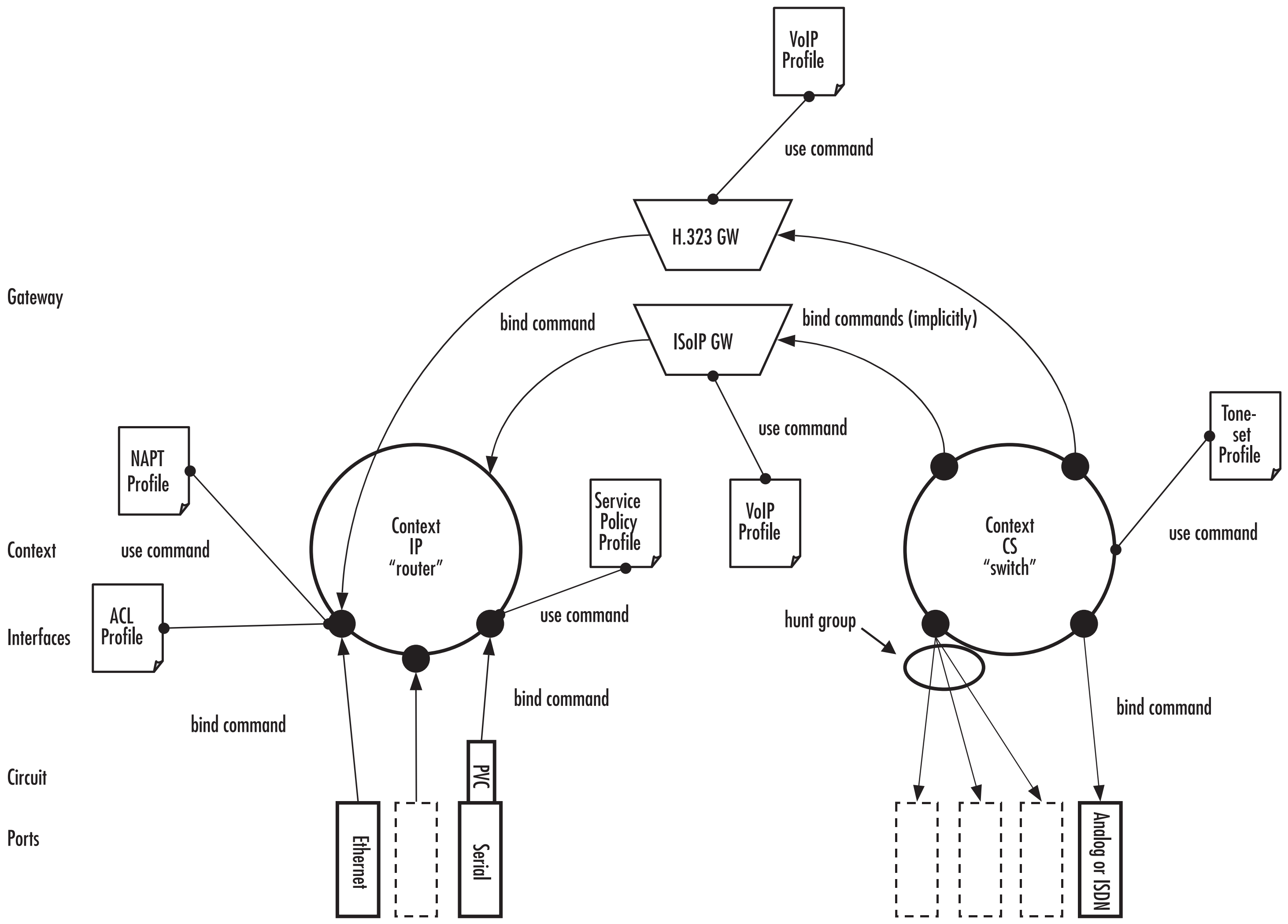


Figure 7. Configuration concept overview

The various elements of a complete SmartNode configuration are shown in figure 7. 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 chapter sections that follow refer to figure 7 on page 31 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. By means of the context concept a separate configuration can be built for newly supported network layer technologies without complicating the configuration methods of existing features. For example, as bridging, ATM or FR switching become available so can a bridging, ATM, or FR context be introduced.

Each context contains a number of *interfaces*, which build the connections to other SmartWare elements and the outside world. Two contexts are shown in figure 7 on page 31: one of type IP named “router”, and 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 circuit-switching context named “switch” can contain number translations, local breakout conditions, and least-cost routing parameters.

Gateway

For the communication between contexts of different types the concept of a *gateway* is introduced. 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 31 shows two gateways, one of type h323 named “h323gw” and one of type ISoIP named “isoipgw”.

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 31, 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 an introduction to the binding concept.

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.

Eight ports are shown in figure 7 on page 31. Three ports are bound directly to an IP interface, one port has a single circuit configured, which is bound to the IP context. Four 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 CS interfaces, bindings are configured in the interface configuration. The binding is created top-down, that is from the interface to the port. CS interfaces can bind one ore more ISDN or PSTN ports. If more than one port is bound, the CS interface is responsible for performing channel hunting on all bound ports. This creates a channel *hunt group*.

Bindings from ports to IP interfaces and from CS interfaces to ISDN ports are shown in figure 7 on page 31.

Profiles and Use commands

Profiles

Profiles provide configuration shortcuts. They contain specific settings which 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 31.

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

The primary user interface to SmartWare is the command line interface (CLI). You can access the CLI either via the SmartNode console port or through a Telnet session. The CLI lets you configure the complete SmartWare functionality, in contrast to other management interfaces (SNMP, HTTP), which are limited to a subset of the functions. CLI commands can be entered on-line 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 40)

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 operator's current working mode is indicated by the CLI prompt. An overview of all command modes is given in figure 8 on page 37 and table 3 on page 38.

Appendix B, "Configuration mode overview" on page 415 contains a detailed overview of all command modes and the CLI commands that are valid in each mode.

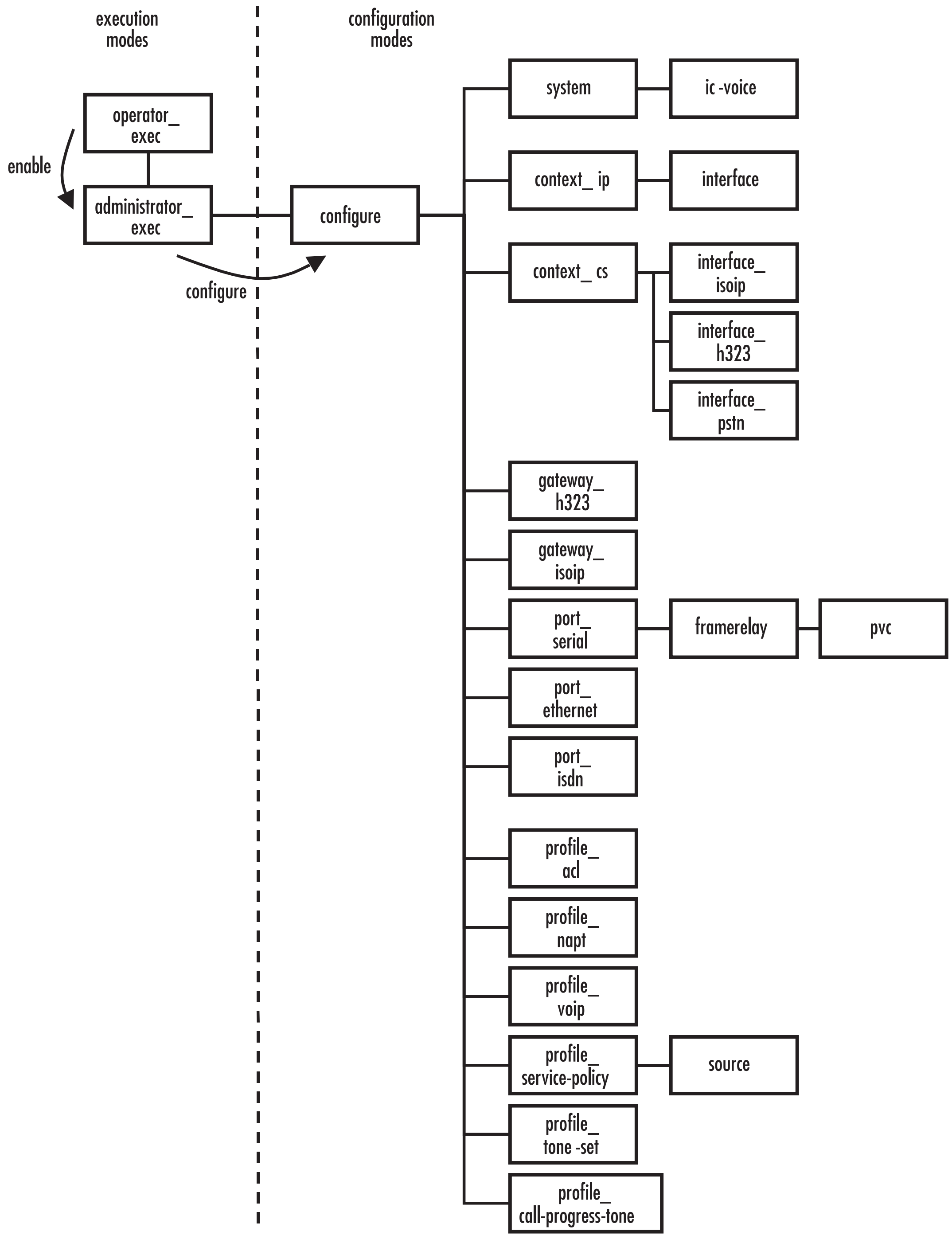


Figure 8. Command line modes

Table 3. Command mode entry and prompts

Mode name	Commands used to access	Command-line prompt
operator exec	(user log-on)	*node*>
administrator exec	**enable** command from operator exec mode	*node*#
configure	**configure** command from administrator exec mode	*node*(config)#
system	**system** command from configure mode	*node*(sys)#
ic-voice	**ic voice <slot>** command from system mode	*node*(ic-voice) [<slot>]#
context_ip	**context ip [router]** command from configure mode	*node*(ctx-ip) [router]#
interface	**interface <name>** command from context_ip mode	*node*(if-ip) [<name>]#
context_cs	**context cs [switch]** command from configure mode	*node*(ctx-cs) [switch]#
interface_pstn	**interface pstn <name>** command from context cs config mode	*node*(if-pstn) [<name>]#
interface_isoip	**interface isoip <name>** command from context cs config mode	*node*(if_isoip) [<name>]#
interface_h323	**interface h323 <name>** command from context cs config mode	*node*(if-h323) [<name>]#
gateway_isoip	**gateway isoip [isoip]** command from configure mode	*node*(gw-isoip) [isoip]#
gateway_h323	**gateway h323 [h323]** command from configure mode	*node*(gw-h323) [h323]#
port_ethernet	**port ethernet <slot> <port>** command from configure mode	*node*(prt-eth) [slot/port]#
port_serial	**port serial <slot> <port>** command from configure mode	*node*(prt-ser) [slot/port]#
framerelay	**framerelay** command from port_serial mode	*node*(frm-rel) [<name>]#
pvc	**pvc <dlci>** command from framerelay mode	*node*(pvc) [<name>]#
port_isdn	**port isdn** command from configure mode	*node*(prt-isdn) [<name>]#
profile_acl	**profile acl <name>**command from configure mode	*node*(pf-acl) [<name>]#
profile_napt	**profile napt <name>** command from configure mode	*node*(pf-napt) [<name>]#
profile_service-policy	**profile policy-map <name>** command from configure mode	*node*(pf-srvpl) [<name>]#
source	**source {class_policy} <name>** command from profile_service-policy mode	*node*(src) [<name>]#
profile_voip	**profile voip <name>** command from configure mode	*node*(pf-voip) [<name>]#
profile_tone-set	**profile tone-set <name>** command from configure mode	*node*(pf-tones) [<name>]#
profile_call-progress-tone	**profile call-progress-tone** command from configure mode	*node*(pf-callp) [<name>]#

System Prompt

For interactive (on-line) sessions the system prompt is of the form:

```
nodename>
```

In the operator exec mode:

```
nodename#
```

In the administrator exec mode and in the different configuration modes:

```
nodename(mode)#
```

Where:

- *nodename* is the currently configured name of the node (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.

Example: the prompt in **configuration mode**, assuming the nodename *SN* is:

```
SN(config)#
```

The CLI commands used to enter each mode and the system prompt that is displayed when you are working within each mode is shown in Table 3 on page 38.

Navigating the CLI

Initial Mode

When you initiate a session you may login with either operator or administrator privileges. Whichever login you use, on starting the CLI is always set to the operator exec (non-privileged exec) mode by default. 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 any changes to the system, the administrator exec (privileged exec) mode must be entered. The **enable** user interface command is used for this purpose. Once in administrator exec mode, all of the system commands are available to you. The enable command is only accessible if you are logged in as an administrator.

Configuration

To make configuration changes the configuration mode must be entered by using the **configure** command in the administrator exec mode. From here the other configuration modes are accessible as diagrammed in the overview in figure 8 on page 37.

Changing Modes

Within any configuration mode, the **exit** command brings the user up one level in the mode hierarchy. 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 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 “**?**” 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 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 entered 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 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 through using the arrow keys.

Command Editing Shortcuts

The SmartWare CLI provides a number of Emacs-style command shortcuts that facilitate editing of the command line. Command editing shortcuts are summarized in table 4 on page 41. 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 4. Command edit shortcuts

Keyboard	Description
Ctrl-p and <up-arrow>	Recall previous command in the command history.
Ctrl-n and <down-arrow>	Recall next command in the command history.
Ctrl-f and <right-arrow>	Move cursor forward one character.
Ctrl-b and <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.

Chapter 4 **Accessing the CLI**

Chapter contents

Introduction

SmartNode products are engineered for remote management and volume deployment. SmartNode management and configuration is therefore based on IP network connectivity. Once a SmartNode is connected to, and addressable in, an IP network then all configuration, management and maintenance tasks can be performed remotely.

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

The ports available for connection and their labels are shown for each SmartNode model 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 35.

IMPORTANT

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

- Accessing via the console port (see page 45)
- Accessing via a telnet session (see page 46)
- Log on to SmartWare (see page 47)
- Selecting a secure password (see page 48)
- Configure operators and administrators (see page 48)
- Displaying the CLI version (see page 50)
- Display account information (see page 50)
- Switching to another log-in account (see page 50)
- Checking identity and connected users (see page 51)
- Ending a Telnet or console port session (see page 52)

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 9). On the host, a terminal emulation application that supports serial interface communication must be used.

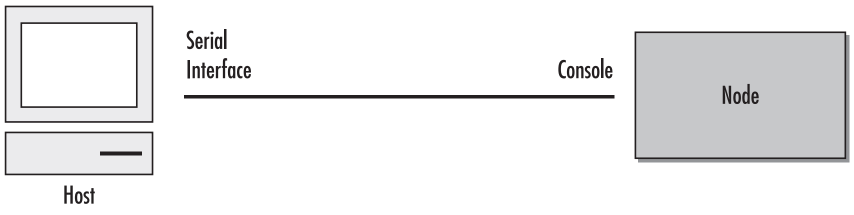


Figure 9. Setup for initial configuration via the console port

Note IP settings do not need to be configured 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 Smart-Node via its network interface. A host can be connected directly to the ETH 1 port (LAN) with a crossover cable (see figure 10, part A) or through an Ethernet hub with two straight cables (see figure 10, part B).

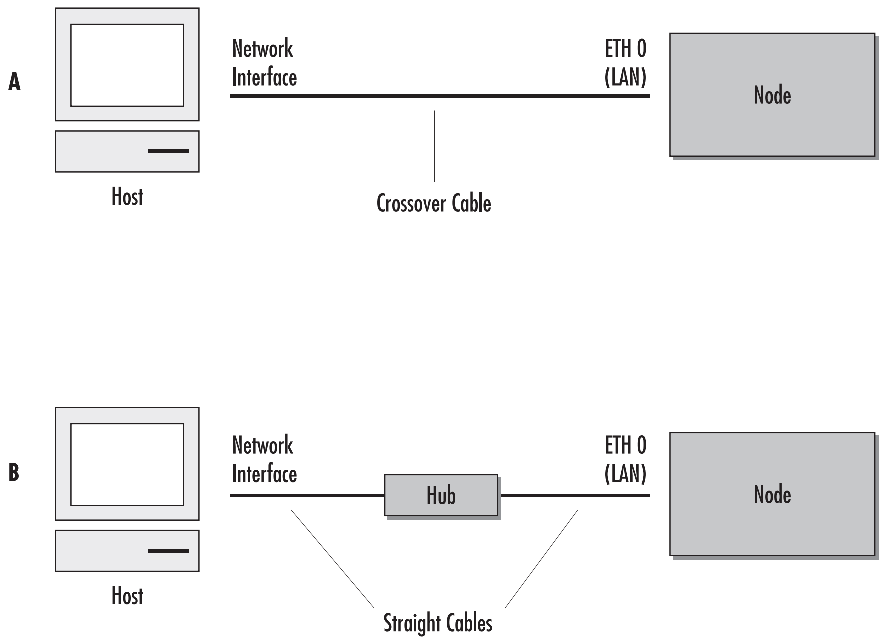


Figure 10. Setup for initial configuration via an Ethernet port

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. The default IP address and network mask of the Ethernet ports of the SmartNode are shown in table 5.

Table 5. 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)	255.255.0.0/24 (formerly 244.255.0.0/16)

Note The default IP addresses listed in table 5 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 the SmartWare release note for more details.

Telnet Procedure

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

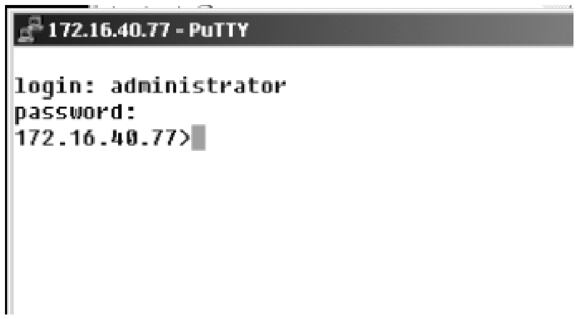
1. Set up the SmartNode as described in the getting started guide included with your SmartNode device.
2. Connect the host (PC) or hub to the ETH 1 (LAN) port of your SmartNode with crossover or straight-thru cables, according to *Appendix A* of the getting started guide included with your SmartNode 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 SmartWare

Accessing your SmartNode via the local console port or via a Telnet session will open 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 will resemble that shown in figure 11. 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.



```
login: administrator
password:
172.16.40.77>
```

Figure 11. 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 11, such as the IP address in the system prompt and window header bar, may be different on your SmartNode device.

IMPORTANT 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 apply 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*

Right now 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, a factory preset administrator account with name *administrator* and an empty password exists in SmartWare. After adding a new administrator account, the factory preset administrator account will be automatically deleted and only the newly created administrator account is available. More than one administrator account can be created, but there has to be at least one administrator account defined. If for some reason the last administrator account is deleted, the factory preset administrator account with name *administrator* and an empty password will automatically be recreated by SmartWare.

Creating an operator account

Operators do not have privileges to run the **enable** command and therefore cannot modify the system configuration. Operators are able to 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 new operator account *name* and password *password*
4	**copy running-config startup-config**	The change made to the running configuration of the SmartNode is saved, so that it will be used following a reload

Example: Create an operator account

The following example shows the commands used 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 decribed 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 new administrator account *name* and password *password*
4	***node(cfg)*#copy running-config startup-config**	Permanently stores the new administrator account parameters.

Example: Create an administrator account

The following example shows the commands used to add a new administrator account with a login name "super" and a matching password of "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 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 : 2.00
```

Displaying account information

The **show** command in SmartWare can be used to display information about existing administrator and operator accounts. This command is not available for an operator account.

Displaying account information is described in the following procedure:

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

To switch from one user account to working in another the **su** command is used. 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, the password of the particular account has to be entered to get privileged access.

Mode: Administrator or operator execution

Step	Command	Purpose
1	***node*>su account-name**	Change to the user user account account-name

Example: Switching to Another Account

The following example shows how to change from your current user account to an administrator account, starting from 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

To display who is logged in or to see more detailed information about users and process states the who command provides this information. Depending on 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
		or
	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 working in.

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 identity of 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

To end a Telnet or console port session you use the **logout** command in the operator or administration execution mode. To confirm the logout command you have to enter “yes” on the dialog line as show in the example below.

Mode: Operator execution

Step	Command	Purpose
1	***node*>logout**	Terminates session after a confirmation by 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

This chapter explains how to establish network-based connections to and from your SmartNode using IP interfaces and Ethernet ports. Configuring basic IP connectivity is carried out in both 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 9, "IP context overview" on page 101, and chapter 10, "IP interface configuration" on page 109.

The chapter includes the following sections:

- IP context selection and basic interface configuration tasks
- Examples (see page 58)

The predefined IP context in SmartWare contains the functionality of a classic IP router. Within the IP context packets are routed between IP interfaces in accordance with 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 basic tasks involved in configuring an IP context, the related interfaces and ports are:

- 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 55)
- Activating the physical port (see page 55)
- Displaying IP interface information (see page 56)
- Deleting IP interfaces (see page 57)

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

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

The 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 *name*, which represents an IP interface. This command also places you in interface configuration mode for the interface *name* you have just created.
3	***node*(if-ip)[*name*]#ipaddress** *ip-address netmask*	Sets the IP address *ip-address* and *netmask* netmask for interface *name*

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 the IP Ethernet encapsulation has to be defined for the related port. It is assumed that you would like to define 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. We assume 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** *slot port*	Enters port configuration mode and selects the Ethernet port port on slot slot, on which IP Ethernet encapsulation shall be used and to which an IP interface shall be bound.
2	***node*(prt-eth)[*slot/port*]#encapsulation ip**	Sets IP Ethernet encapsulation for port port on slot slot
3	***node*(prt-eth)[*slot/port*]#bind interface** *name* **router**	Binds interface name to port port on slot slot to the IP context named router, which is the IP router context

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

We assume you would like to set IP encapsulation for 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 all the settings for the IP interface are completed the physical port has to be activated. The SmartWare default status for any port is disabled. In the 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 CLOSED for the current state.

```
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 the shutdown status of the port has to be removed. That means, the state of the port has to be changed 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 *slot port*	Enters port configuration mode and selects the Ethernet port *port* on slot *slot*, which is to be activated
2	***node*(prt-eth)[*slot/port*]#no shutdown**	Activates the physical port *port* on slot *slot* for operation

Example: Activating the physical port

We assume 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

Information for all the configured IP interfaces can be displayed by 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

Displaying IP interface information 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

Deleting an existing interface in the IP context is often necessary. The procedure illustrated 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 *name*

Example: Delete IP interfaces

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

First list the existing interfaces:

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

Now delete the interfaces named “external” with the **no interface** command with the interface name as argument:

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

Finally 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. The relation between the IP interface *lan* and the Ethernet port 0 on slot 0 is shown in figure 12. The configuration procedure below starts in the operator execution mode:

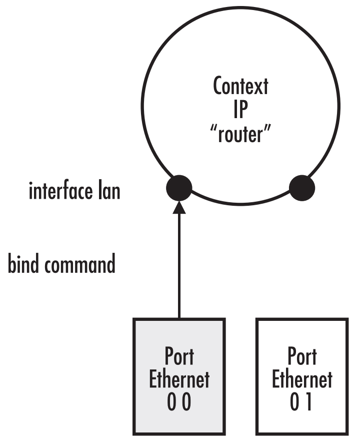


Figure 12. Relation between IP Interface *lan* and Ethernet port 0 on slot 0

First the context IP mode is selected for the required IP interface configuration.

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

After that a new interface *lan* is created, 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
```

Next the Ethernet port 0 on slot 0 is selected, the medium is set to 10 Mbps in half-duplex mode, and IP encapsulation for this port is chosen.

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

Afterwards the just defined interface *lan* is bound to the Ethernet port, and then the port is activated.

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

As a final point the configuration settings are stored 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

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 has to be stored in the persistent region of the memory. Driver software images contain software that also has to be stored in the persistent region of the memory and are used for optional PMC interface cards.

This chapter includes the following sections:

- Memory regions in SmartWare
- Boot procedure and bootloader (see page 64)
- Factory configuration (see page 72)
- System image handling task list (see page 73)

All Patton SmartNode devices are shipped with a default system image, which is stored in the persistent flash memory of the SmartNode at the factory. The system image contains the application image and driver software images that together build SmartWare. In addition a factory configuration is loaded to the SmartNode at the factory, which sets initial SmartWare parameters. Therefore the user is neither has to load a system image not the factory configuration to the SmartNode prior using it.

Your own operational configuration files are stored in the SmartNode flash memory, and copies may also be stored on a remote server. Transferring configuration files between the flash memory and a remote server requires the use of the Trivial File Transfer Protocol (TFTP). The TFTP server must be accessible through one of the SmartNode IP interfaces. TFTP is not possible over the console interface.

In the following sections the focus is on SmartWare memory regions and the software components that can be copied within the memory or moved between a TFTP server and the memory of the SmartNode. Since SmartWare uses a specific vocabulary in naming those software components, refer appendix A, "Terms and definitions" on page 407 to ensure that you understand the concepts.

Memory regions in SmartWare

The SmartNode memory used by SmartWare is divided into several regions as shown in figure 13 on page 63. A remote TFTP server is used for up- or downloading configurations, application and driver software images to or from the SmartNode's memory. In the SmartWare command syntax, the file path of a file on the TFTP server that is used for image upload or download is prefixed with *tftp:* followed by the absolute file path starting from the root directory of the TFTP server.

The flash memory stores data contained in it persistently and is made up of two logical regions called *flash:* and *nvram:*, which are used as follows:

- The application image, a bootloader image and one or more driver software images have to be stored in the logical region *flash:* of flash memory.
- Configuration files have to be stored in the logical region *nvram:* 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 *nvram*.

The factory configuration is read-only, and is contained in the persistent memory in the logical region *nvram:* 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 the SmartWare terminology. A dedicated user-specific configuration has to 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 has to be stored as "default-config" in the logical region *nvram:* of flash memory. Any configuration stored within the persistent memory in the logical region *nvram:* can be copied to a remote server using TFTP.

Since configurations are not executable from persistent memory, any configuration that is to be used has to 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 flash memory as the "running-config" into the volatile memory *system:* of the SmartNode. The volatile memory *system:* is a logical region within the random access memory (RAM) of the SmartNode.

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

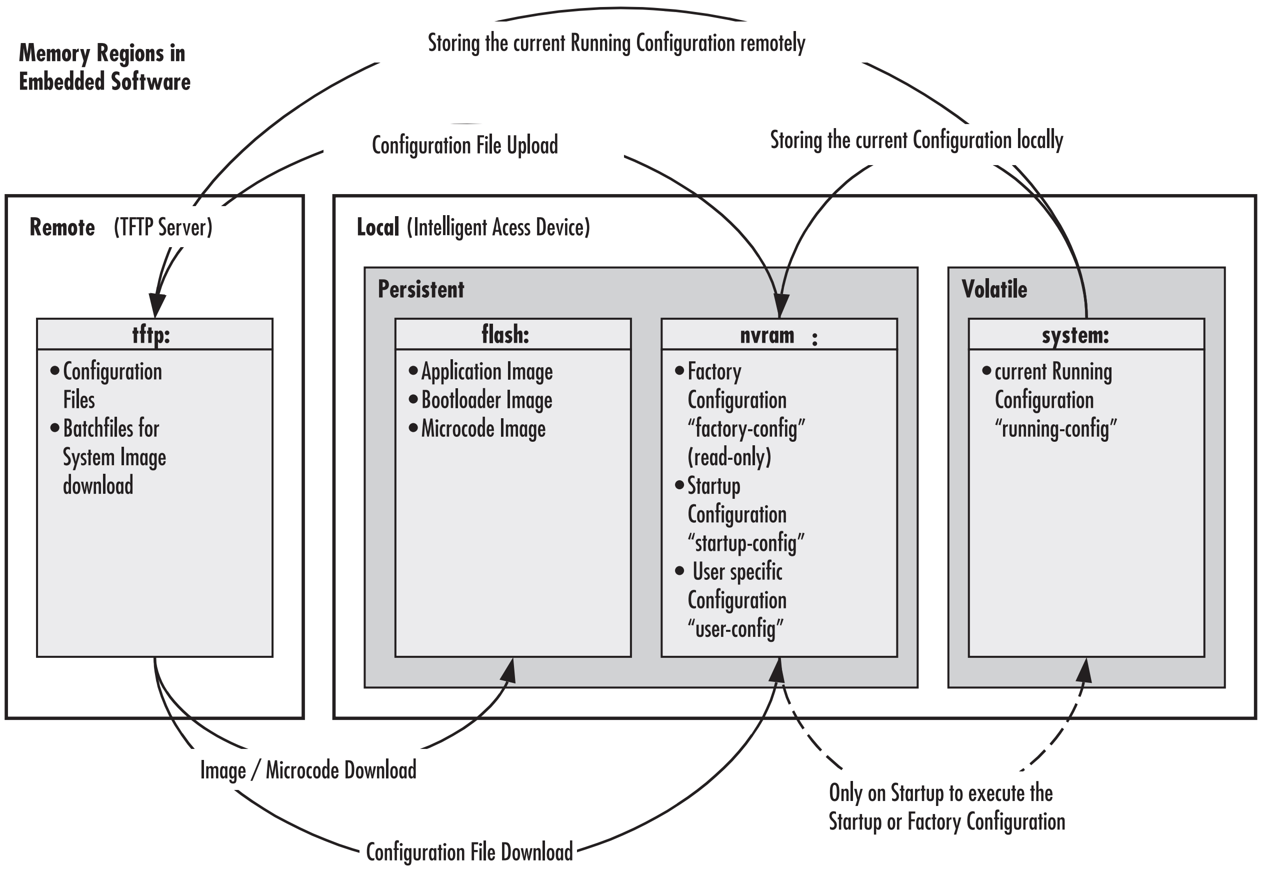

Memory Regions in Embedded Software
Storing the current Running Configuration remotely
Configuration File Upload
Storing the current Configuration locally
Remote (TFTP Server)
Local (Intelligent Acess Device)
Persistent
Volatile
tftp:
• Configuration Files
• Batchfiles for System Image download
flash:
• Application Image
• Bootloader Image
• Microcode Image
nvram :
• Factory Configuration "factory-config" (read-only)
• Startup Configuration "startup-config"
• User specific Configuration "user-config"
system:
• current Running Configuration "running-config"
Image / Microcode Download
Configuration File Download
Only on Startup to execute the Startup or Factory Configuration

Figure 13. SmartNode memory regions logically defined in smartware

Boot procedure

During a normal boot procedure of a SmartNode the bootstrap application checks the persistent memory in the logical region *nvram*: for an application image. Following that the application image is executed, that is 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 flash memory as "running-config" into the volatile memory *system:* and is used to parameterize SmartWare. The boot procedure is illustrated in figure 14.

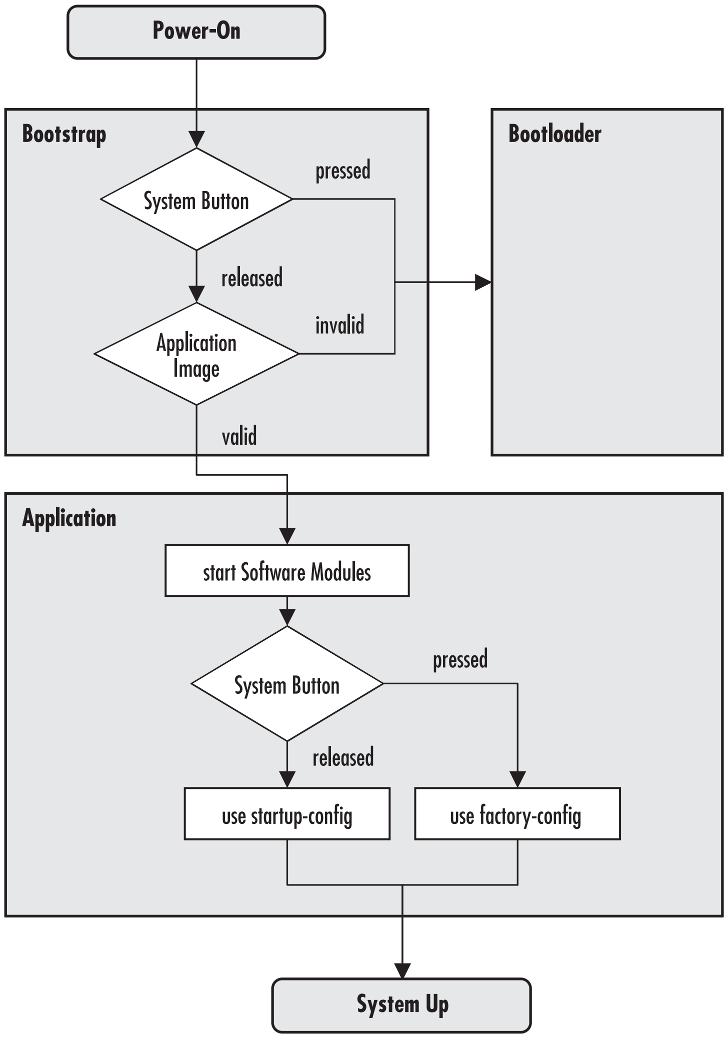


Figure 14. Boot Procedure

There are two situations during bootstrap during which the bootloader takes control. The bootstrap application checks the status of the Reset button (not available for SN4xxx) on the back plane of the SmartNode, and if the system button has been pressed it launches the bootloader. The bootloader is also launched if a valid application image is not available.

The bootloader ensures that basic operations, network access and downloads are possible in cases 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 will still use the previous application image.

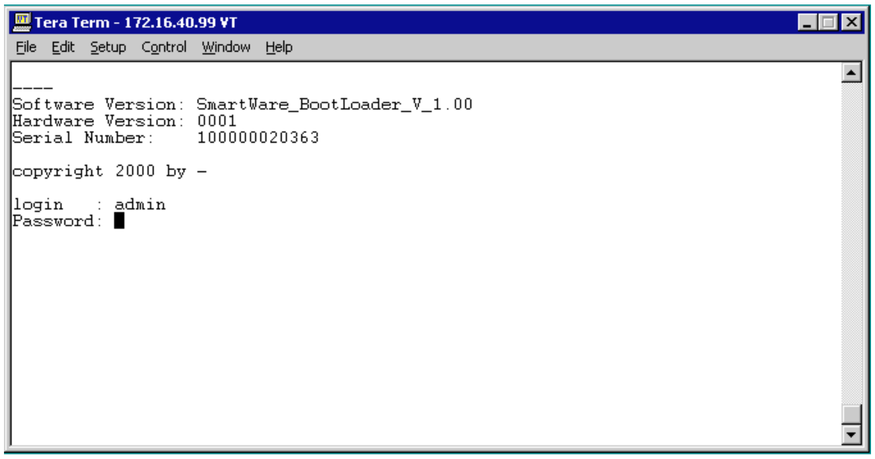
If the application image is valid it is started, and SmartWare is brought into operation module by module. During this system initialization phase (when the message *Press reset button to restore factory defaults...* appears on the console screen), the status of the reset button on the back plane of the SmartNode is checked. If the button has been pressed the factory configuration is loaded into the volatile memory and is used to parameterize 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 SmartWare.

Bootloader (for SmartNode 1000 and 2000 Series)

Start Bootloader and login

To start the Bootloader explicitly, power the SmartNode *while* pressing the reset button. All front LEDs will light up. Keep the reset button pressed until the BRI/Ethernet-LEDs on SmartNode 1x00 and the ACT-LED on SmartNode 2x00 extinguish. When the bootloader is started, 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 15 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 the default gateway, which they had before starting the bootloader.


Tera Term - 172.16.40.99 VT
File Edit Setup Control Window Help

```
----
Software Version: SmartWare_BootLoader_V_1.00
Hardware Version: 0001
Serial Number:    100000020363

copyright 2000 by -

login    : admin
Password:
```

Figure 15. Login display

Main shell and domains

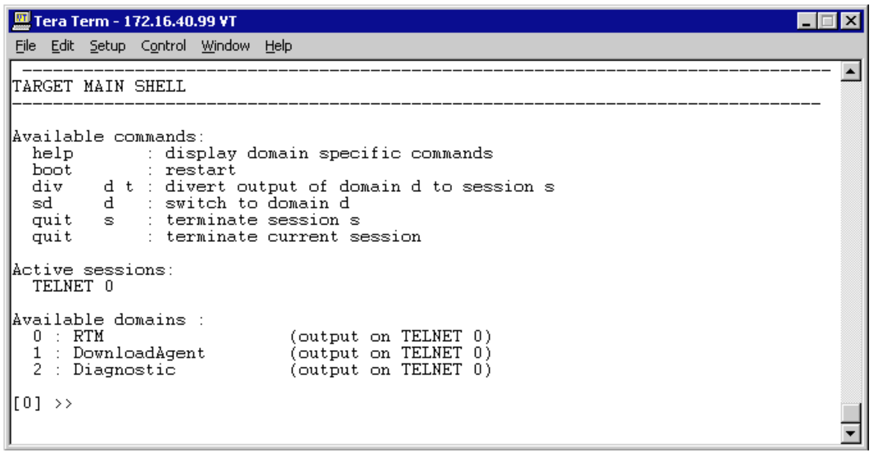
After login, you enter the Main Shell (see Figure 0-4). It offers you to select from three available Domains:

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

The available command set is shown in table 6.

Table 6. 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	The output of domain d is diverted to another session s, i.e. to Telnet interfaces.
sd d	Switches to another domain. The available domains are displayed in the main menu by entering ?. d = 0 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

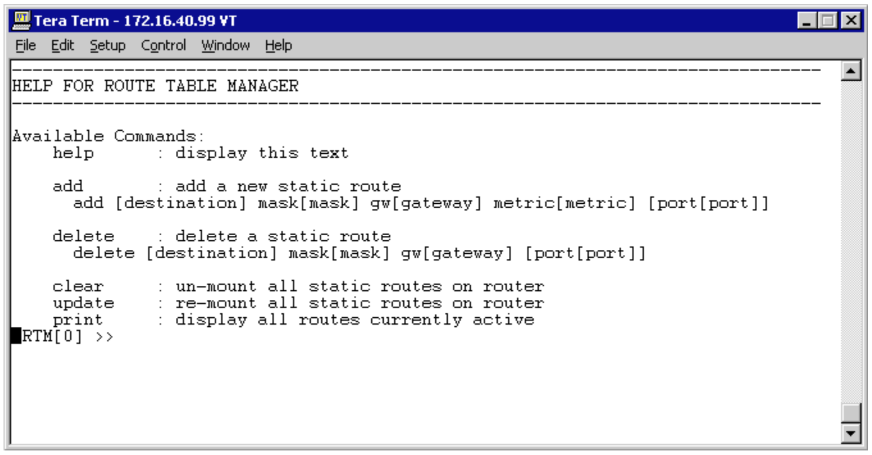
```
--------------------------------------------------------------------------------
TARGET MAIN SHELL
-------------------------------------------------------------------------------

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 16. 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 17).


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 17. 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
```

A route may be deleted by using the command 'delete'.

For example:

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

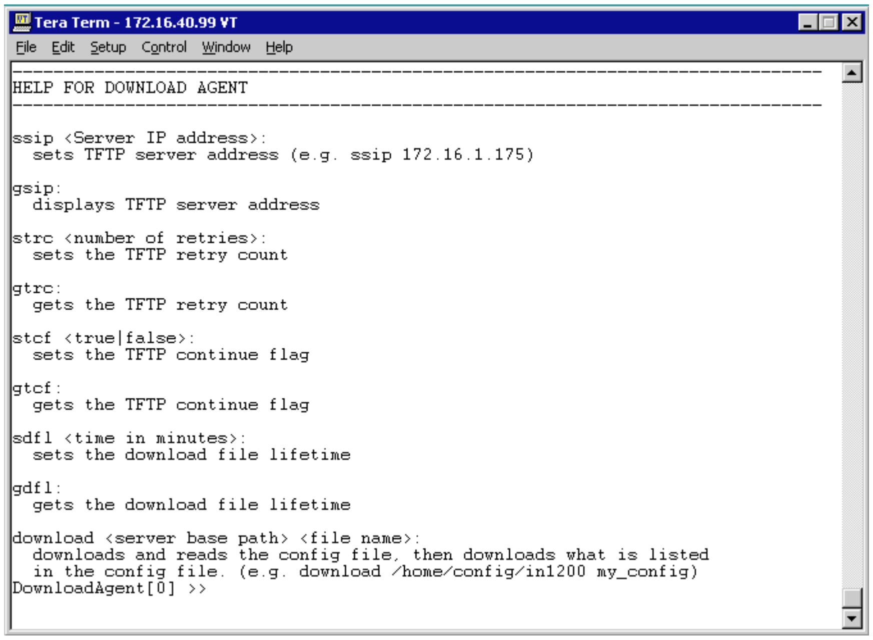
To deactivate/activate all static routes, use the commands 'clear' followed by 'update'.

A list of all routes currently active is displayed 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 18).


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

sdfl <time in minutes>:
  sets the download file lifetime

gdfl:
  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 18. 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 (‘sdfl’, ’gdfl’)

The command ‘download’ is used to download an application image or a configuration file from the TFTP server, for example:

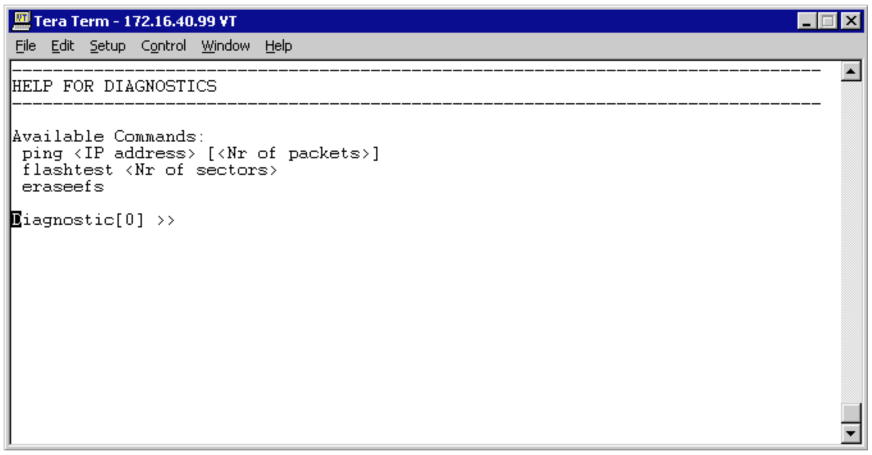
```
download /SmartWare/Sn1xxx/Vx/R2.10/BUILD21215 b
```

where ‘/SmartWare/...’ is the path to 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 19).


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 19. Diagnostic Display

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

The various sectors of the SmartNode flash memory can be tested with the command ’flashtest’.

To delete the contents of the EFS the command ’eraseefs’ is used.

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 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>
```

Enter ‘help’ to get an overview of the available commands. The cursor keys (up, down, left, right) are not working, use CTRL-N (for up) and CTRL-P (for down) instead. Overwriting and inserting does not work either. 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** *local_ip_address* **[/***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. *mask_len* 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** *gateway*	Sets the IP address of the default gateway.
3 optional	**RedBoot> ping -h** *tftp-server_ip_address*	Test connectivity to the TFTP server.
4	**RedBoot> load -r -v -h** *host* **-b** *base_address file_name*	Downloads an application image into the volatile memory (RAM) from where the SmartNode could directly execute it. *host*: IP address of the TFTP server *base_address*: memory location where to store the application image. Use default address 0x1800100 *file_name*: 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	Delete the first application image. Reply with 'y' to the confirmation request.
6	RedBoot> fis create	Store the downloaded application image to the permanent memory (flash:). Reply with 'y' to the confirmation request.
7	RedBoot> fis list -l	Check 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	Start the application image that was downloaded to 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 was last configured by the Bootloader itself or by an other 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 the 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>/b1 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 serial link

The Bootloader supports protocols 'X-Modem' and 'Y-Modem' 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 I ymodem } -b** *base_address*	Downloads an application image into the volatile memory (RAM) from where the SmartNode could directly execute it. 'xmodem' or 'ymoden': Specify the protocol to be used, X-Modem or Y-Modem *base_address*: memory location where to store the application image. Use default address 0x1800100 Execute the above RedBoot command first, then start the transfer from terminal program with the command 'Send file via X-Modem' (or similar).
5	RedBoot> fis delete -n 1	Delete the first application image. Reply with 'y' to the confirmation request.
6	RedBoot> fis create	Store the downloaded application image to the permanent memory (flash:). Reply with 'y' to the confirmation request.
7	RedBoot> fis list -l	Check whether the image has been successfully stored, whether it is the desired Release and Build, and whether it is valid.
8	RedBoot> go	Start 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 a serial link at only 9600 Baud is used.

Factory configuration

SmartNodes are delivered with a *factory configuration* stored in the logical region *nvram:* of the memory. It is used to initially parameterize the network and component settings of SmartWare, which make sense at the very beginning. Moreover if a SmartWare is malfunctioning resetting to the initial state is done by reloading the factory configuration. The factory configuration consists of default settings for the IP networking subsystem.

As soon as a user-specific configuration is created and stored as startup configuration, the factory configuration will no longer used but it remains in the persistent memory. At any time during operation of a SmartNode it is possible to switch back to the factory configuration. See section "Boot procedure" on page 64 and section "Start-up with factory configuration" on page 70 for information on how to restore the factory-configuration.

IMPORTANT

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 Flash memory (see page 74)
- Copying driver software from a network server to Flash memory (see page 75)

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 R2.00 BUILD22031
Supplier         :
Provider         :
Subscriber       :

Information for Slot 0:
SN2300 (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 above the system image file contains the application software that runs SmartWare; it is loaded in the flash memory at the Patton Electronics Co. factory. Since most of the voice and data features of the SmartNode are defined and implemented within 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 has to 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 only have to download new SmartWare application software 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 a special download script file has to be used. This script file defines how the system image file is to be handled and where it is to be stored. You cannot download any system image file without an appropriate script file.

Each line in 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
#the next line deletes the whole embedded file system
*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!

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

Mode: Administrator execution

Step	Command	Purpose
1	***node***(cfg)# **copy tftp://***node-ip-address***/***b* **flash:**	Downloads the script file *b* from the TFTP server at address *node-ip-address* 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 Flash memory

The following example shows how to download the system image file and command line syntax definition 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 system image file and command line syntax definition file are 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 driver software that is to be downloaded to 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 driver software image file a special download script file must be used.

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 the driver software image file is to be handled and where it is to be stored.

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

Mode: Administrator execution

Step	Command	Purpose
1	***node*(cfg)# copy tftp://***node-ip-address***/***b* **flash:**	Downloads the script file *b* from the TFTP server at address *node-ip-address* 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 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

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 within the software modules that are performing specific functions of the SmartNode. Some aspects of configuration file management are also described in this chapter. Refer to chapter 6, "System image handling" on page 61 for more information.

This chapter includes the following sections:

- Factory configuration (see page 80)
- Configuration file handling task list (see page 81)

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

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 using the **enable** command to enter administrator execution mode. Then you must switch to the configuration mode by typing the command **configure**. Once in configuration mode you can 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. Configuration files can be copied 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 43 for information concerning access to the CLI.

In the following sections the emphasis is on SmartWare memory regions and on software components that can be copied within the memory or be up/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 407 to ensure that you understand the concepts. Refer to chapter 6, "System image handling" on page 61 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, during operation of a SmartNode, you enter a command using the CLI of SmartWare, you alter the running configuration accordingly. In other words you are modifying a live, in-service system configuration.

Figure 20, shows the characteristics of a configuration file. This configuration is stored on a TFTP server in the file *SN2300_001.cfg* for later download to the SmartNode *SN*. The command syntax is identical for commands entered by the use of the CLI and commands contained in configuration files. For better comprehension SmartWare allows comments within 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 2.00
  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

# CS context configuration
  context cs switch
  no number-prefix national
  no number-prefix international
  use tone-set-profile default
  called-party rtab 201 dest-interface telecom-operator
  called-party rtab 202 dest-interface telecom-operator
  no shutdown

# 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

  # interface used to access the PSTN telecom operator
    interface pstn pstn-operator
    routing dest-interface h323
    bind port 1 0

  # interface used to access the VoIP telecom provider
    interface h323 voip-provider
    routing dest-table rtab
    remoteip 172.19.33.60

  # H.323 gateway primarily used
    gateway h323
    codec g711alaw64k 10 20
    codec g711ulaw64k 10 20
    faststart
    no ras
    gatekeeper-discovery auto
    bind interface lan router
    use voip-profile default
    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 20. Sample configuration file

Each configuration file that is 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 file names used to represent the factory configuration, startup configuration and running configuration, which are *factory-config*, *startup-config*, and *running-config*. Refer to appendix A, "Terms and definitions" on page 407 to learn more about configuration file types.

Factory configuration

Patton SmartNodes are delivered with a *factory configuration* in the logical region *nvram:* of the SmartNode that is used to initially parameterize the network and component settings of SmartWare that are most useful when starting initially. Moreover, if a SmartWare is malfunctioning, resetting to the initial state is possibly reloading the factory configuration. The factory configuration consists of:

- Default settings for the IP networking subsystem
- Default settings for H.323 and ISoIP gateway subsystem

- Default settings for the quality of service subsystem

As soon as 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. At any time during the operation of a SmartNode it is possible to switch back to the factory configuration. The restoration procedure for restoring the default settings is described in the getting started guide included with your SmartNode device.

IMPORTANT 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 so as 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 using the **configure** command; see section "Modifying the running configuration at the CLI" on page 87 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 82)
- Replacing the startup configuration with a configuration from Flash memory (see page 83)
- Copying configurations to and from a remote storing location (see page 85)
- Replacing the startup configuration with a configuration downloaded from TFTP server (see page 86)
- Displaying configuration file information (see page 87)
- Modifying the running configuration at the CLI (see page 87)
- Modifying the running configuration offline (see page 88)
- Deleting a specified configuration (see page 89)

Copying configurations within the local memory

Configuration files may be copied within the local memory in order to switch between different configurations. Remember the different local memory regions in SmartWare as shown in figure 21.

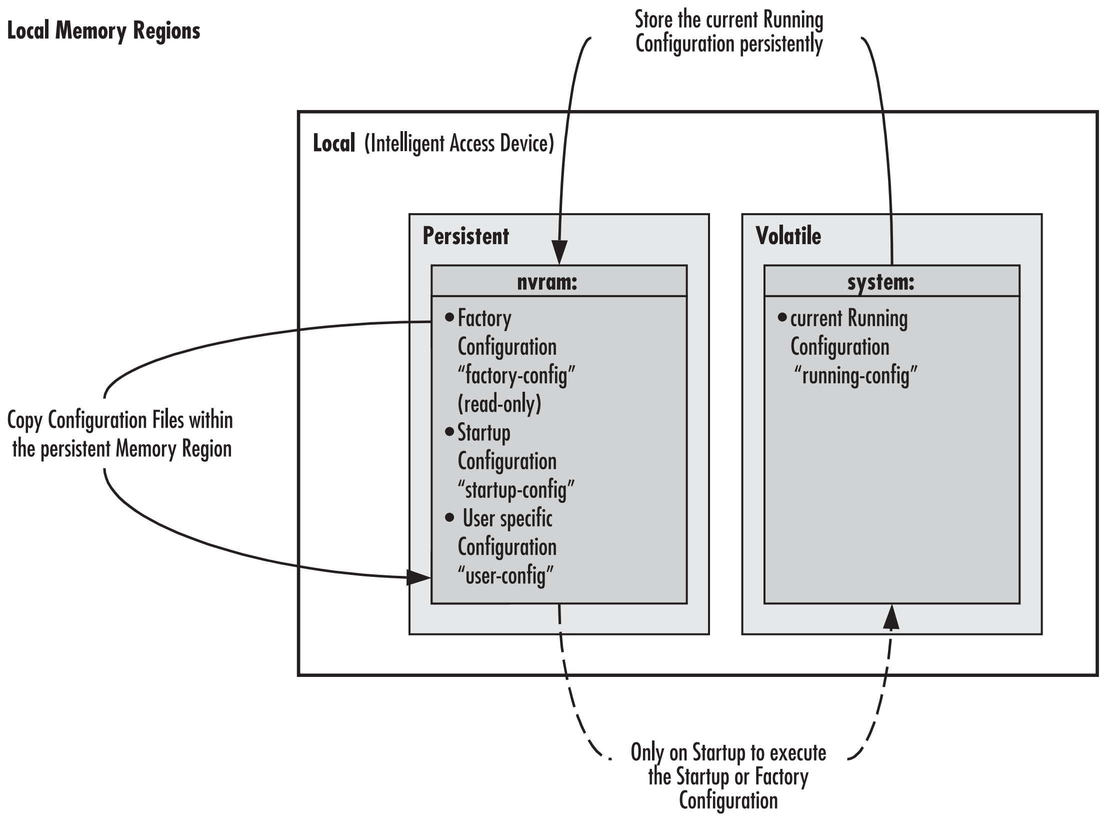

Local Memory Regions
Store the current Running Configuration persistently
Local (Intelligent Access Device)
Persistent
Volatile
nvram:
• Factory Configuration "factory-config" (read-only)
• Startup Configuration "startup-config"
• User specific Configuration "user-config"
system:
• current Running Configuration "running-config"
Copy Configuration Files within the persistent Memory Region
Only on Startup to execute the Startup or Factory Configuration

Figure 21. Local memory regions in SmartWare

In the majority of cases, the interactively modified running configuration known as the *running-config*, which is to be found in the volatile memory region *system:*, is copied to the persistent memory region *nvram:*. This running config is stored under the name *startup-config* and replaces the existing startup configuration.

The current running configuration can be copied to the persistent memory region *nvram*: under a user-specified name, if that configuration is to be preserved.

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

As shown in figure 21 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, within the same memory region any configuration file needs a unique name so for example it is not possible to have two configurations 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 files names for which the specification of the memory region is optional, namely the files *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 *source-name* as target configuration file *target-name* in local memory.

Example: Backing up the startup configuration

The following example shows how to make a backup copy of the startup configuration. In a first the startup-config is copied under the name backup within the flash memory region nvram:.

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

Replacing the startup configuration with a configuration from Flash memory

The startup configuration is replaced by a configuration that is already present in the flash memory, by copying it to that area of the flash memory where the startup configuration is to be 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 *new-startup* 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 using the **copy** command.

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

The following example shows how to overwrite and therefore replace the persistent startup configuration in the flash memory of a SmartNode with the configuration contained in the file *new-startup* already present in the flash memory.

1. Replace the current startup configuration, using the **copy** command, into the flash memory area where the startup configuration has to be 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 R2.00 BUILD220               #
# 2001-10-25T09:20:42                    #
# Generated configuration file           #
#----------------------------------------#

  cli version 2.00
  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 22 on page 85. 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.

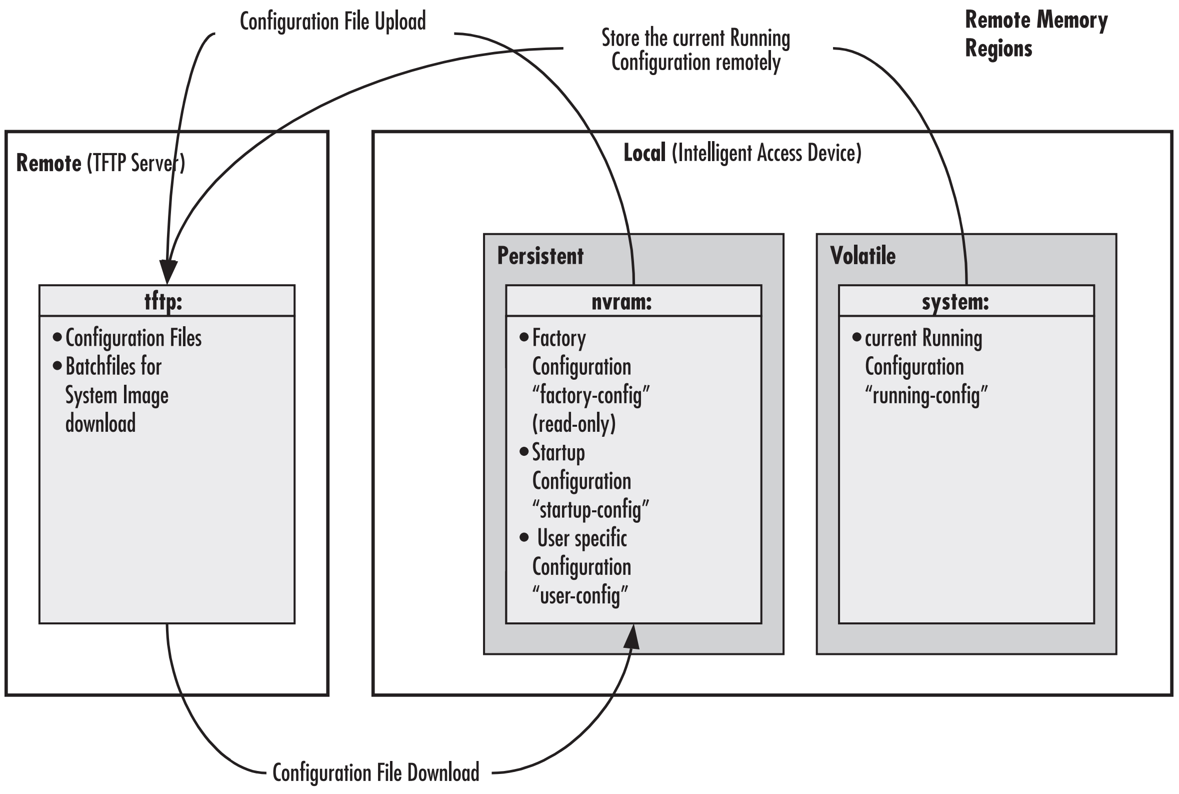

Configuration File Upload
Store the current Running Configuration remotely
Remote Memory Regions
Remote (TFTP Server)
Local (Intelligent Access Device)
Persistent
Volatile
tftp:
• Configuration Files
• Batchfiles for System Image download
nvram:
• Factory Configuration "factory-config" (read-only)
• Startup Configuration "startup-config"
• User specific Configuration "user-config"
system:
• current Running Configuration "running-config"
Configuration File Download

Figure 22. 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 which is to be found 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*, to the persistent memory region *nvram:* of the target device. After this the startup configuration is transferred to the TFTP server from where it can be distributed to other SmartNode devices, which 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, the startup-configuration is replaced by downloading a configuration from the TFTP server into the flash memory area where the startup configuration has to be stored.

Mode: Administrator execution

Step	Command	Purpose
1	**node(cfg)# copy tftp://***ip-address/new-startup* **nvram:startup-config**	Downloads the configuration file *new-startup* from the TFTP server at address *ip-address* replacing the existing persistent startup configuration. This progress is visualized with a counter, counting up from 0 to 100% according to the downloaded amount of the file size. If the download should fail an error message *% File Transfer - Get failed* is displayed.

Example: Sample configuration download from TFTP server

The following example shows how to overwrite and therefore replace the persistent startup configuration in the flash memory of a SmartNode 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 the startup configuration is to be stored.

```
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 R2.00 BUILD22031                                     #
# 2001-10-25T09:20:42                                            #
# Generated configuration file                                   #
#----------------------------------------------------------------#

  cli version 2.00
  snmp community public rw


framerelay
  exit

SN#
```

Displaying configuration file information

This procedures describes how to display information about configuration files

Mode: Administrator execution

Command	Purpose
show nvram:	List of 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

The 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 into 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 using the CLI, the shell executes the commands as you enter them.

When you log-in to a SmartNode using the CLI all commands entered directly modify the running configuration, which is located in the volatile memory region *system:* (or RAM) of your SmartNode. Remember that this memory is—as its name suggests—volatile, therefore if your modifications shall be permanent you have to copy the configuration to the persistent memory. In most cases you will store it as the upcoming startup configuration and therefore store it 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. As a final the SmartNode has to be restarted using the **reload** command.

Mode: Administrator execution

Step	Command	Purpose
1	***node*#configure**	Enters administrator configuration mode
2	Enter all necessary configuration commands.	

Step	Command	Purpose
3	***node*(cfg)#copy running-config startup-config**	Saves the running configuration file as upcoming startup configuration
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, a SmartNode's running configuration may be stored on a TFTP server and there edited and saved. Since the SmartNode is acting as a TFTP client, all file transfer operations are initiated from the SmartNode.

First the running configuration, named *running-config*, has to be uploaded from the SmartNode to the TFTP server. After that the configuration file located on the TFTP server gets edited using any regular text editor. Followed by downloading the configuration back to the SmartNode as upcoming startup configuration and therefore store it in the persistent memory region *nvram:* under the name *startup-config*. Finally the SmartNode has to be restarted using the **reload** command to activate the changes.

Note Consider that a user-specific configuration file does not manipulate any function of SmartWare until it is copied to—and therefore replaces—the configuration file *startup-config*. Downloading configuration files to flash memory using a name other then *startup-config* is typically useful to activate any configuration changes or to store configuration for backup purposes in the flash memory of the SmartNode.

Mode: Administrator execution

Step	Command	Purpose
1	***node*#copy running-config tftp://***node-ip-address/current-config*	Uploads the current running configuration as file current-config to the TFTP server at address node-ip-address. 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 current-config 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 current-config from the TFTP server at address node-ip-address into the persistent memory region nvram: using the name startup-config. This progress is visualized with a counter, counting up from 0 to 100% according to the downloaded amount of the file size. If the download should 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 the commands used 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 will be 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. Following the modified configuration file *current-config* is downloaded from the TFTP server, at IP address 172.16.36.80, to the SmartNode's persistent memory region *nvram:* using the name *startup-config*. Finally the SmartNode has to be restarted.

```
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 procedures 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 *name* from 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 "isoip-config" is to be deleted, since it is no longer used.

1. First the command **show nvram:** is used with to list all available configurations.

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

2. The configuration named *minimal* has to be deleted explicitly.

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

3. The command **show nvram:** is entered again 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

This chapter describes parameters that report basic system information to the operator or administrator, and their configuration.

There are basic SmartWare parameters that need to be established when first setting up a new system. The administrator needs to define the system's hostname, set the location of the system, provide reference contact information, and set the clock.

In addition basic management tasks such as checking the CRC of configuration files, displaying the currently running SmartWare commands, moving SmartWare commands back into foreground, setting the system banner, enabling the embedded web server, and other task of system character are described in this chapter.

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.

- Setting system information (see page 92)
- Setting the system banner (see page 94)
- Setting time and date (see page 94)
- Display clock information (see page 95)
- Display time since last restart (see page 95)
- Configuring and starting the web server (see page 96)
- Determining and defining the active CLI version (see page 96)
- Restarting the system (see page 97)
- Displaying the system event log (see page 97)
- Controlling command execution (see page 98)
- Displaying the checksum of a configuration (see page 99)

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 Co. 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 Co. 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 Co. enterprise-specific MIB supplier object.

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#system contact** *information*	Sets the contact information to *information*
2	***node*(cfg)#system hostname** *information*	Sets the hostname to *information*
3	***node*(cfg)#system location** *information*	Sets the location information to *information*
4	***node*(cfg)#system provider** *information*	Sets the provider information to *information*
5	***node*(cfg)#system subscriber** *information*	Sets the subscriber information to *information*
6	***node*(cfg)#system supplier** *information*	Sets the supplier information to *information*

Note If system information has to be formed out of more than one word the information is enclosed by 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 23). 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 23. System banner with message to operators

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#banner** *message*	Sets the message for the system banner to *message*

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** *yyyy-mm-ddThh:mm:ss*	Sets the system clock to *yyyy-mm-ddThh:mm:ss*

Note SmartWare includes an integrated SNTP client, which allows synchronization of time-of-day and date to a reference time server. Refer to chapter 19, "SNTP client configuration" on page 223 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, which 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 lang {*de* | *en*}**	Sets the language to either German (de) or English (en)
2	***node*(cfg)#webserver port number**	Sets the listening port number in the 1 to 65535, default port number for 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 lang 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
2	***node*(cfg)#cli version** *version.revision*	Selects the active CLI version in the form version.revision

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 : 2.00
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.

IMPORTANT

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.

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.
2001-12-12T13:46:20 : LOGWARNING : Authentication failure.
2001-12-12T13:46:31 : LOGWARNING : Authentication failure.
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 *jobid* 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 Telnet sessions

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

Mode: System

Step	Command	Purpose
1	[*name*] **(sys)#terminal height**	Configures the terminal height.
2	[*name*] **(sys)#terminal idle-time-logout**	After 30 minutes without user input the current Telnet session is closed. If longer session periods are required (logging/debugging) this command allows to increase the session timeout up to 136 years.
3	[*name*] **(sys)#terminal more**	Enables pausing of display for commands which produce more output than the current terminal window can display at once.
4	[*name*] **(sys)#terminal width**	Configures the terminal width.

Chapter 9 IP context overview

Chapter contents

Introduction

This chapter outlines the SmartWare *Internet protocol (IP) context*, together with its related components. You will get the fundamental understanding on how to set up your SmartNode to make use of IP related services.

In the following sections configuration steps necessary to put together certain IP services are illustrated, together with the references to the related chapters that explain the issue in more details.

Understanding the information given in the following chapters requires that you carefully read to the end of this chapter. Prior proceeding with this chapter make that you feel comfortable with the underlying SmartWare configuration concept by reading chapter 2, "Configuration concepts" on page 29.

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. In a first approximation the IP context performs the same function as a standalone IP router. Every context is defined by a name; therefore the IP context is named *router* for default. This IP context may contain interface static routes, RIP parameters, NAPT, QoS and access control profiles, and related ISoIP or H.323 gateways

In figure 24, the IP context with all its related elements is contained within the area on the left, which has a gray fill. On the right side the related CS context is shown, 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 24 with gray lines instead of black.

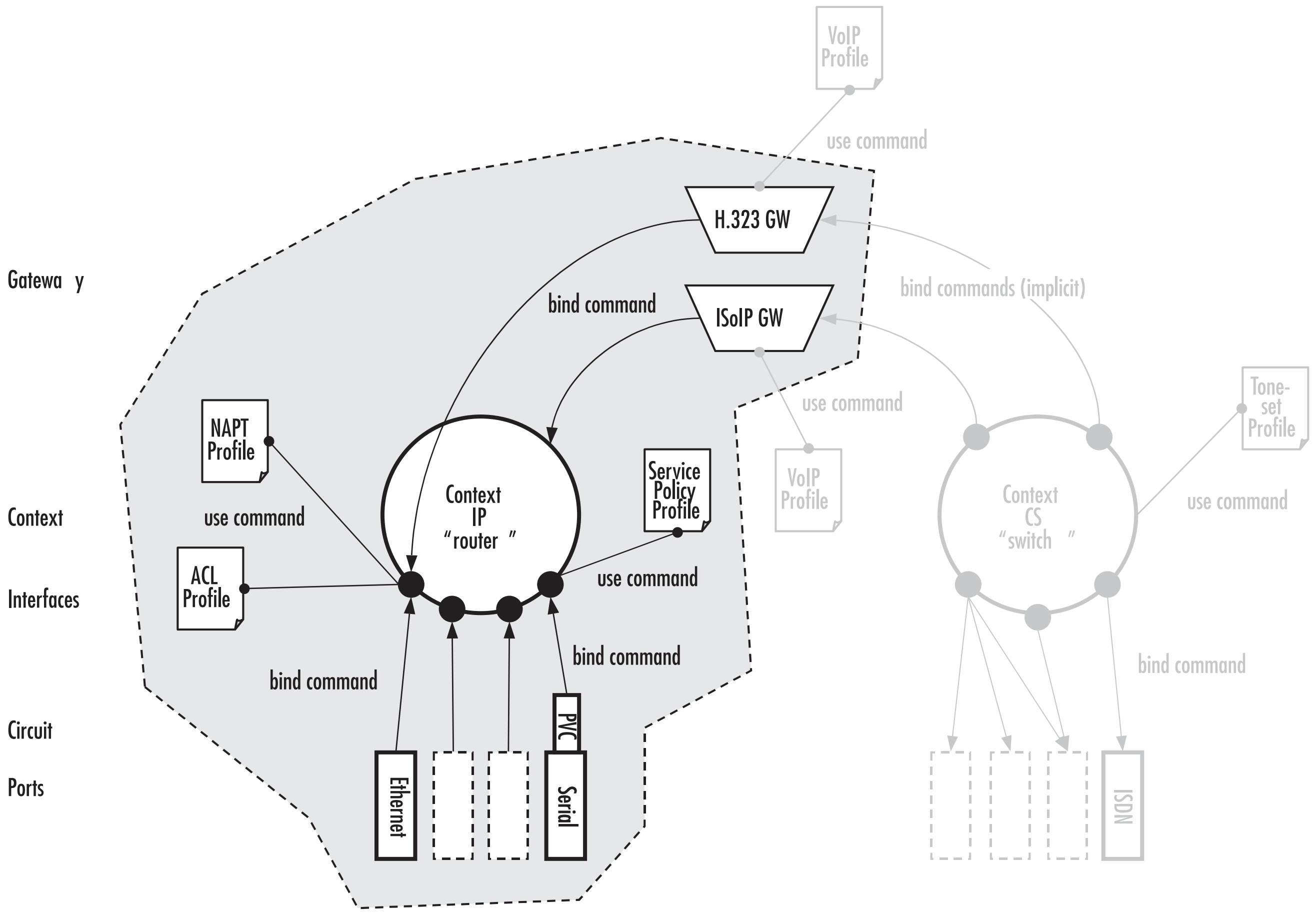


Figure 24. IP Context and Related Elements

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 packets have to be handled 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. For that reason it will not give you all the details of a configuration task, but guides you to the chapters in which you will find the full description.

- All the information you need to configure an IP Interface is to be found in chapter 10, "IP interface configuration" on page 109.
- Information regarding network address port translation (NAPT) in chapter 11, "NAT/NAPT configuration" on page 119.

- If you need to configure a physical port, chapter 12, "Ethernet port configuration" on page 127 or chapter 14, "Serial port configuration" on page 159 may prove helpful.
- To set up the IP router contained within SmartWare, 15, "Basic IP routing configuration" on page 177 on page 177 and 16, "RIP configuration" on page 183 on page 183 give you the necessary inside information.
- Related to network security requirements, 17, "Access control list configuration" on page 193 provides essential knowledge.
- Finally, if your network shall provide better service to selected network traffic, chapter 13, "Link scheduler configuration" on page 137 will help you with getting in-depth knowledge about quality of service (QoS) management with SmartWare.

The basic tasks involved in IP context configuration are described in the following sections. 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 set up on 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 104)
- Configuring Ethernet and serial ports (see page 105)
- Creating and configuring IP interfaces (see page 105)
- Configuring NAPT (see page 106)
- Configuring static IP routing (see page 106)
- Configuring RIP (see page 106)
- Configuring access control lists (see page 107)
- Configuring quality of service (see page 107)

Planning your IP configuration

Network connection considerations are provided for several types of physical ports types in the following subsections. Drawing a network overview diagram displaying all neighboring IP nodes and serial connected elements is recommended. 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 central H.323 gatekeeper
- IP addresses of central PSTN gateway for H.323 and ISoIP based calls

- IP addresses of 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 per 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 you are connecting to is a DCE providing a clock signal on the synchronous serial interface.
- Type of connector, male or female, required connecting 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 12, "Ethernet port configuration" on page 127 or for a serial port to chapter 14, "Serial port configuration" on page 159.

Creating and configuring IP interfaces

Today 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 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 self-explanatory names should be used which depict the use of the interface. An example is using 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. During the operation of a SmartNode it is possible to move an IP interface to another physical port, e.g. from an Ethernet to a serial port. For that reason it would be more than misleading, if an interface holds a name like “eth0”, but actuality is assigned to a serial port. Therefore it is in your interest to decouple a logical interface from a physical port, by giving names to interfaces that describe their usage and not the physical constitution.

As for any IP interface 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 10, “IP interface configuration” on page 109.

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. Therefore NAPT allows 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 made invisible to the outside world. Configuring NAPT is done by creating a profile that is afterwards used on an explicit IP interface. In the terminology of SmartWare an IP interface *uses* a NAPT profile, as shown in figure 24 on page 103.

For information and examples on how to configure Network Address Port Translation (NAPT) refer to chapter 11, “NAT/NAPT configuration” on page 119.

Configuring Static IP Routing

SmartWare allows defining 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 15, “Basic IP routing configuration” on page 177.

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 (that is, 16), the network destination is considered unreachable.

For information and examples on how to configure Routing Information Protocol (RIP) refer to chapter 16, "RIP configuration" on page 183.

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 configure access control lists refer to chapter 17, "Access control list configuration" on page 193.

Configuring quality of service (QoS)

In SmartWare the link scheduler allows the definition of QoS profiles for network traffic on a certain interface, as shown figure 24 on page 103. 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

The SmartWare QoS features described in chapter 13, "Link scheduler configuration" on page 137 address these diverse and common needs.

Chapter 10 IP interface configuration

Chapter contents

Introduction

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

Within the Patton 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 the SmartWare. For higher-layer protocols to become active, a physical port or circuit must be bound to an interface. Therefore it is possible to bind an IP interface physically to an Ethernet, SDSL or Frame Relay port, 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 110)
- Deleting an IP interface (see page 111)
- Setting the IP address and netmask (see page 112)
- ICMP message processing (see page 112)
- ICMP redirect messages (see page 112)
- Router advertisement broadcast message (see page 113)
- Defining the MTU of the interface (see page 114)
- Configuring an interface as a point-to-point link (see page 114)
- Displaying IP interface information (see page 115)
- Testing connections with the **ping** command (see page 115)

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** *name*	Creates the new interface *name*, 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 specific configuration parameters for IP interface *name* can be entered

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.

Delete an existing interface in the IP context is often necessary.

Mode: Context IP

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

Example: Delete IP interfaces

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

First list the existing interfaces:

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

Now delete the interfaces named “eth3” with the **no interface** command:

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

Finally 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. To set the IP address to *ip-address* and the network mask to *netmask* or enable IP processing for IP interface *name* without assigning an explicit IP address, use the **ipaddress** interface configuration command. The **ipaddress** command offers the following options:

unnumbered	Enables IP processing on an interface without assigning an explicit IP address to the interface.
ip-address	Specifies the IP address of the subscriber in the form A.B.C.D.
netmask	Specifies the network mask in the form A.B.C.D. A network mask of at least 24 bits must be entered; that is, 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 22, "DHCP Configuration".

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

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

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* of 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. 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 following ICMP message processing features:

- ICMP redirect messages
- Router advertisement broadcast message

ICMP redirect messages

Routes are sometimes less than optimal. For example, it is possible for the router to be forced to resend a packet through the same interface on which it was received. If the router resends a packet through the same interface on which it was received, 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.

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

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

Mode: Interface

Step	Command	Purpose
1	***node***(**ctx-ip)[router]#interface** *name*	Selects interface *name* for ICMP message processing configuration
2	***node***(**if-ip)[***name***]#icmp redirect { accept | send}**	Enables sending or accepting of ICMP redirect messages

Example: ICMP redirect messages

The following example shows how to configure ICMP messages processing to accept ICMP redirect 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 redirect accept
```

Router advertisement broadcast message

This message configures the behavior of the router when receiving an ICMP router solicitation messages, 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 for 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 for ICMP router solicitation messages nor periodically.

Mode: Interface

Step	Command	Purpose
1	***node***(**ctx-ip)[router]#interface** *name*	Selects interface *name* for ICMP message processing configuration
2	***node***(**if-ip)[***name***]# icmp router-discovery**	Enables sending of 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

Mode: Interface

Step	Command	Purpose
1	***node*(ctx-ip)[router]#interface** *name*	Selects interface *name* for ICMP message processing configuration
2	***node*(if-ip)[*name*]#mtu size**	Sets the IP MTU packet size to *size* for the interface *name*. A possible value for MTU packet size has to be in the range from 48 to 1500.
Step 3 (optional)	***node*(if-ip)[*name*]#tcp adjust-mss { rx|tx } { mtu | *mss* }**	Limits to the MSS (Maximum Segment Size) in TCP SYN packets to *mss* 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	***node*(cfg)#context ip router**	Selects the IP router context
2	***node*(ctx-ip)[router]#interface** *name*	Selects the defined interface *name* for configuration
3	***node*(if-ip)[*name*]#point-to-point**	Configures interface ifname as point-to-point link

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

The following example shows how to define 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 IP information for all interfaces

example: displaying ip interface information

The following example shows how to display IP information for all interfaces 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 *ip-address*

When using **ping** for fault isolation, it should first be run on the respective SmartNode interface, to verify that the local LAN or WAN interface is up and running. Then hosts and gateways further and further away should be “pinged”. 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 in calculating 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 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 network host.

Mode: Either operator or administrator execution

Step	Command	Purpose
1	***node*>traceroute** *ip-address*	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*, use the **no** command as following demonstrated in IP context configuration mode.

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 use 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 11 NAT/NAPT configuration

Chapter contents

Introduction

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

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 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 will apply the terminology defined in RFC 2663.

SmartWare provides four different types of NAT/NAPT:

- Dynamic NAPT (Cisco-terminology: NAT Overload)
- Static NAPT (Cisco-terminology: Port Static NAT)
- Dynamic NAT
- Static NAT

These types of NAT/NAPT can be combined 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 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. Is a connection 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 25 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.

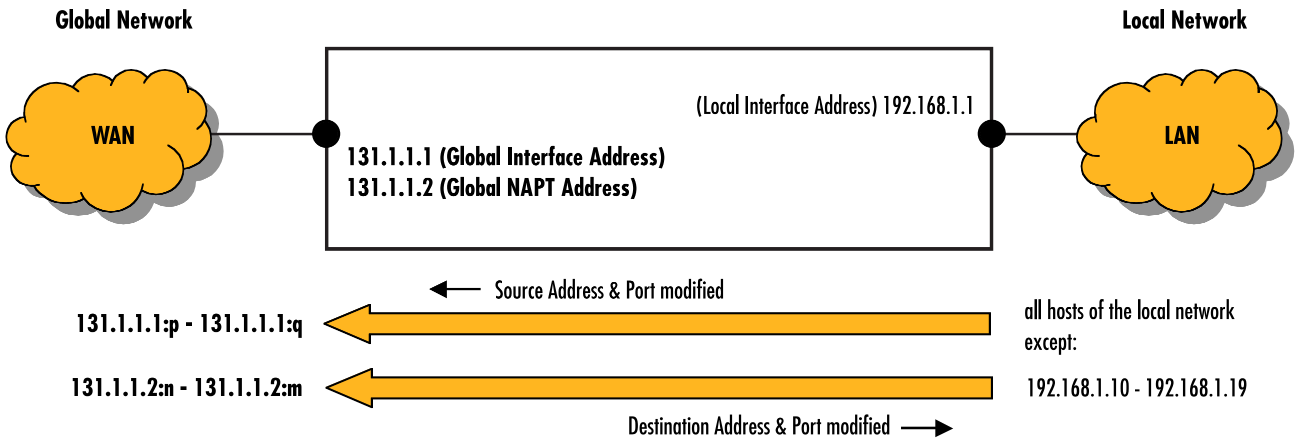

Global Network
Local Network
WAN
LAN
(Local Interface Address) 192.168.1.1
131.1.1.1 (Global Interface Address)
131.1.1.2 (Global NAPT Address)
Source Address & Port modified
131.1.1.1:p - 131.1.1.1:q
all hosts of the local network except:
131.1.1.2:n - 131.1.1.2:m
192.168.1.10 - 192.168.1.19
Destination Address & Port modified

Figure 25. 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.

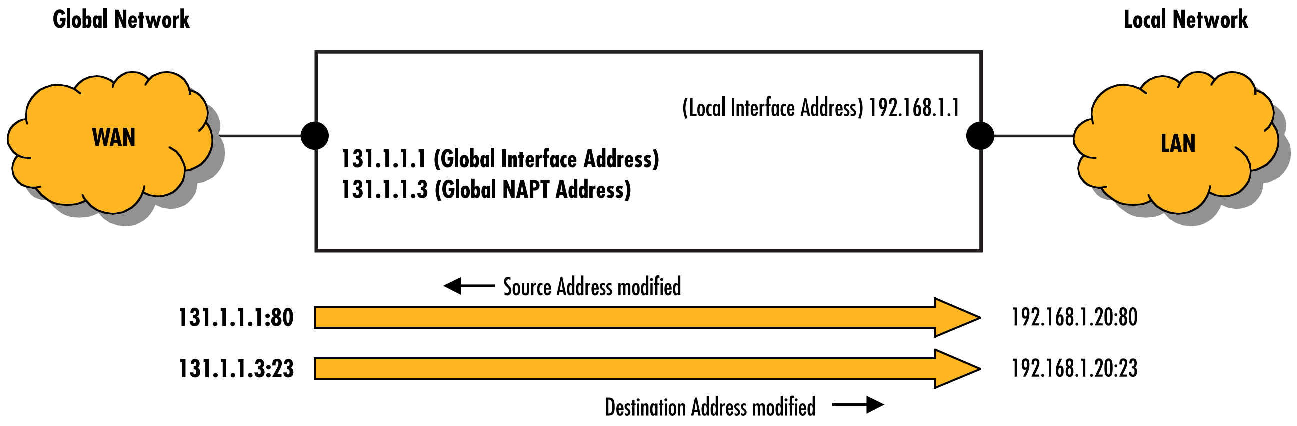

Global Network
Local Network
WAN
LAN
(Local Interface Address) 192.168.1.1
131.1.1.1 (Global Interface Address)
131.1.1.3 (Global NAPT Address)
Source Address modified
131.1.1.1:80
192.168.1.20:80
131.1.1.3:23
192.168.1.20:23
Destination Address modified

Figure 26. 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 of dynamic NAT entry for the reverse path. Is a connection 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".

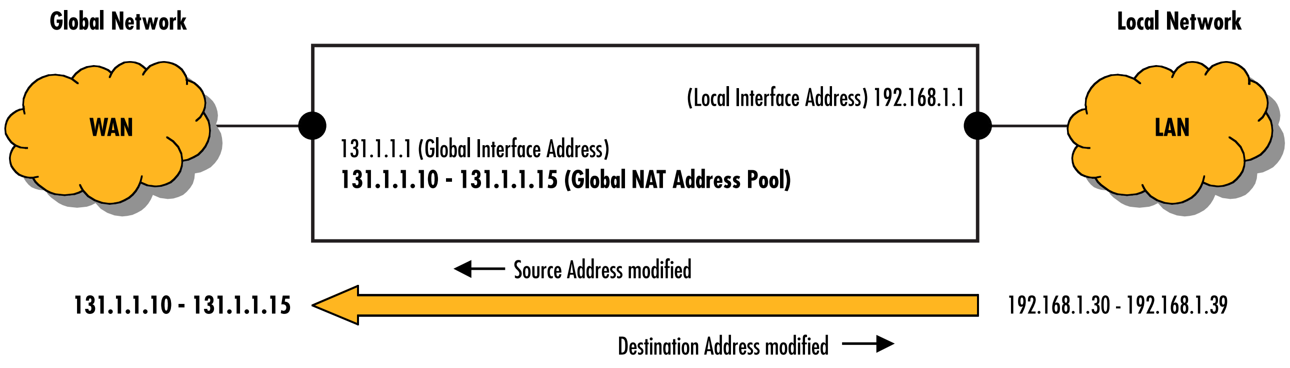

Global Network
Local Network
WAN
LAN
(Local Interface Address) 192.168.1.1
131.1.1.1 (Global Interface Address)
131.1.1.10 - 131.1.1.15 (Global NAT Address Pool)
Source Address modified
131.1.1.10 - 131.1.1.15
192.168.1.30 - 192.168.1.39
Destination Address modified

Figure 27. 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".

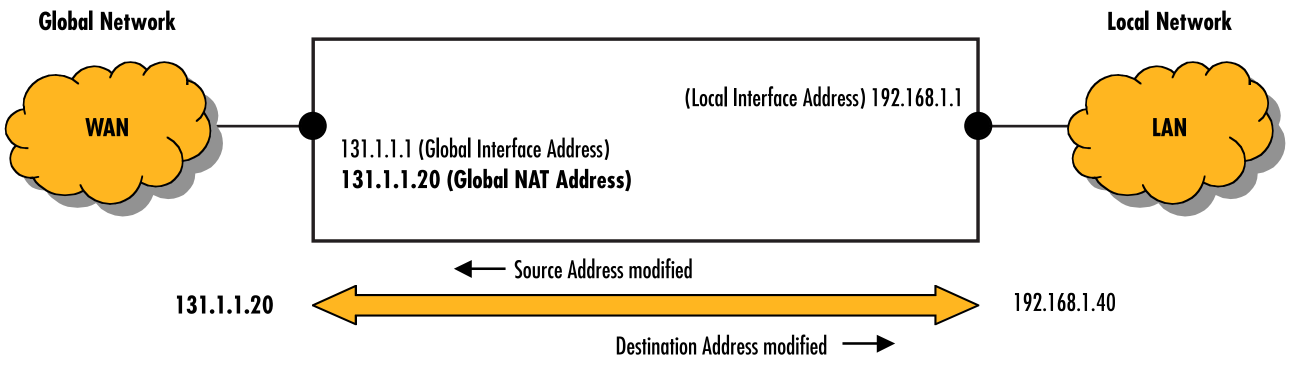

Global Network
Local Network
WAN
LAN
(Local Interface Address) 192.168.1.1
131.1.1.1 (Global Interface Address)
131.1.1.20 (Global NAT Address)
Source Address modified
131.1.1.20
192.168.1.40
Destination Address modified

Figure 28. 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 123)
- Activating NAT/NAPT (see page 123)
- Displaying NAT/NAPT configuration information (see page 125)

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	*node*(cfg)#**profile napt** *name*	Creates the NAPT profile *name* and activates the basic behavior of the Dynamic NAPT
Step 2 (optional)	*node*(pf-napt)[*name*]#**range** *local-ip-range-start local-ip-range-stop global-ip*	Configures and activates the enhanced behavior of the Dynamic NAPT: *local-ip-range-start* and *local-ip-range-stop* define the subset of local hosts that use the global NAT address *global-ip* 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)	*node*(pf-napt)[*name*]#**static { udp | tcp }** *local-ip local-port* [*global-ip*] [*global-port*]	Creates a Static NAPT entry: *local-ip / local-port* is mapped to *global-ip / global-port*. If *global-port* is omitted, *local-port* is used on both sides. If *global-ip* is omitted, the global address is the address of the global interface. (max. 20 UDP and 20 TCP entries)
Step 4 (optional)	*node*(pf-napt)[*name*]#**range** *local-ip-range-start local-ip-range-stop global-ip-start global-ip-stop*	Configures and activates the Dynamic NAT: *local-ip-range-start* and *local-ip-range-stop* define the subset of local hosts that use an address from the global NAT address pool to access to global network. *global-ip-start* and *global-ip -stop* 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)	*node*(pf-napt)[*name*]#**static** *local-ip global-ip*	Creates a Static NAT entry: *local-ip* is mapped to *global-ip*. (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 discontinued.

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 120.

```
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	*node*(cfg)#**context ip router**	Selects the IP router context
2	*node*(ctx-ip)[router]#**interface** *name*	The NAPT profile shall be used on the interface *name*
3	*node*(if-ip)[*name*]#**use profile napt** *profile*	Defines that the NAPT profile *profile* shall be used on the interface *name*

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	*node*(cfg)#**show profile napt**	Displays the available NAPT profiles
Step 2	*node*(cfg)#**show profile napt** *name* or *node*(cfg)#**show napt interface** *name*	Displays the NAPT profile *name* or Displays the NAPT profile bound to the IP interface *name*

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 12 Ethernet port configuration

Chapter contents

Introduction

This chapter provides an overview of Ethernet ports and describes the tasks involved in configuring Ethernet ports through the Patton 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 129)
- Configuring medium for an Ethernet port (see page 129)
- Configuring Ethernet encapsulation type for an Ethernet port (see page 130)
- Binding an Ethernet port to an IP interface (see page 130)
- Selecting the frame format for an Ethernet port (see page 131)
- Configuring layer 2 CoS to service class mapping for an Ethernet port (advanced) (see page 132)
- Closing an Ethernet port (see page 134)

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 7.

Table 7. 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
	FXS	0	0	FXS 0/0
			…	…
			7 (max)	FXS 0/7

Since a port must be configured unambiguously, choose the appropriate expansion slot and port number. The number and type of available ports depends 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** *slot port*	Enters Ethernet port configuration mode for the interface on *slot* and *port*
2	***node*(prt-eth)[*slot/port*]#medium (10 | 100 | auto} (half | full)**	Configures interface on *slot* and *port* 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 supports 10 Mbps half-duplex
- SN4xxx: Ethernet port 0/1 pretends to be a 100-Mbps port. However the speed is internally limited to 10 Mbps.

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 1000, 2000, or 4000 series 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** *slot port*	Enters Ethernet port configuration mode for the interface on *slot* and *port*
2	***node*(prt-eth)[***slot/port***]#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. At the time of 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 29 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.

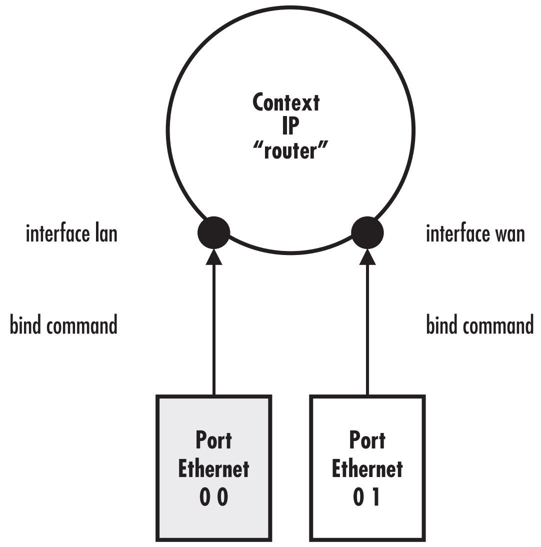


Figure 29. Binding of an Ethernet port to an IP interface

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

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#port ethernet** *slot port*	Enters Ethernet port configuration mode for the interface on *slot* and *port*
2	***node*(prt-eth)[***slot/port***]#bind interface** *name* **router**	Binds the Ethernet port to the already existing IP interface *if-name*

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 either of 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** *slot port*	Enters Ethernet port configuration mode for the interface on *slot* and *port*
2	***node***(**prt-eth)[***slot/port***]#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 real-time and delay sensitive services such as VoIP traffic to be transported across the network, the SmartWare 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** *slot port*	Enters Ethernet port configuration mode for the interface on *slot* and *port*
2	***node***(**prt-eth)[***slot/port***]#cos {default | rx-map | tx-map }**	Selects the layer 2 CoS to service class mapping

If the frame format is set to standard, the **cos default** command value defines which class of service has to be used 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 SmartWare-specific service class value. Each conversion is stored as a mapping table entry. Therefore 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** *slot port*	Enters Ethernet port configuration mode for the interface on *slot* and *port*
2	***node*(prt-eth)[*slot/port*]#cos rx-map** *layer 2 class of service value as service class value*	Adds a receive mapping table entry, which converts a *layer 2 class of service* into a *service class value*

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 1000, 2000, or 4000 series device.

```
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 SmartWare-specific service class value into a Layer 2 CoS to service class value. Each conversion is stored as a mapping table entry. Therefore 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** *slot port*	Enters Ethernet port configuration mode for the interface on *slot* and *port*
2	***node*(prt-eth)[*slot/port*]#cos tx-map** *service class value* **as** *layer 2 class of service value*	Adds a transmit mapping table entry, which converts a *service class value* into a *layer 2 class of service*

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 1000, 2000, or 4000 series device.

```
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** *slot port*	Enters Ethernet port configuration mode for the interface on *slot* and *port*
2	***node*(prt-eth)[*slot/port*]#shutdown**	Disables Ethernet port on *slot* and *port*

The **no** prefix causes the port to be opened together with the interface to which the port is bound.

Example: Disabling an Ethernet port

The following example shows how 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 13 **Link scheduler configuration**

Chapter contents

Introduction

This chapter describes how to use and configure the SmartWare Quality of Service (QoS) features. Refer to chapter 17, "Access control list configuration" on page 193 more information on the use of access control lists.

This chapter includes the following sections:

- Quick references (see page 139)
- Packet Classification (see page 141)
- Assigning bandwidth to traffic classes (see page 144)
- Link scheduler configuration task list (see page 141)

QoS in networking refers to the capability of the network to provide a better service to selected network traffic. While this chapter focuses on how SmartWare supports QoS in customer premises applications, the entire end-to-end path through the network must be QoS-aware to ensure the best possible service.

When a SmartNode acts as an access router and voice gateway, the access link is the point at which intelligent use of the scarce resources really makes a difference in preventing a bottleneck. This chapter shows you how to configure the SmartWare to make best use of the access link.

The procedures include:

- Minimizing voice delay
- Reducing delay for interactive applications
- Controlling coexistence of other applications

The SmartWare QoS features are only applicable on transmitted data packets, i.e. towards the uplink interface, but not for data packets received at your SmartNode.

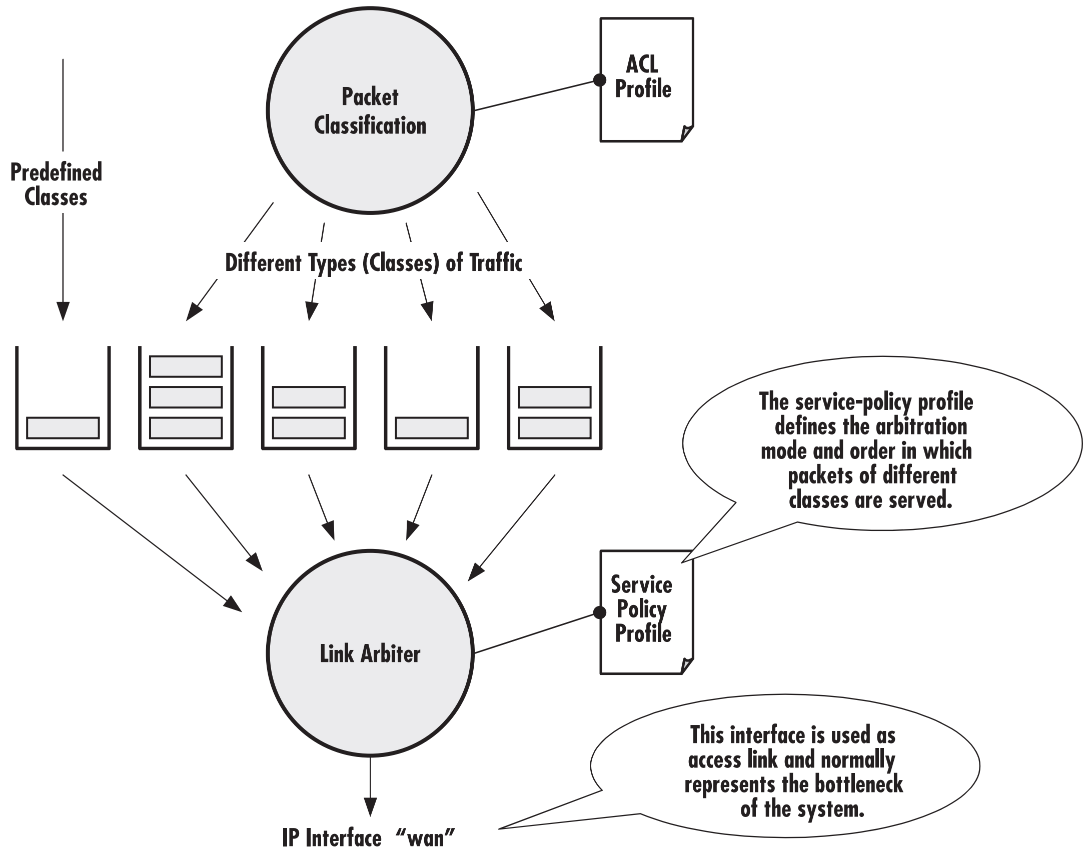

Packet Classification
ACL Profile
Predefined Classes
Different Types (Classes) of Traffic
The service-policy profile defines the arbitration mode and order in which packets of different classes are served.
Service Policy Profile
Link Arbiter
This interface is used as access link and normally represents the bottleneck of the system.
IP Interface "wan"

Figure 30. Service-Policy Profile devoted to an Interface

The link arbiter controls outbound network traffic according to a service-policy profile devoted to a certain interface, as shown in figure 30. Take into account, that such an interface normally represents the very bottleneck in the data transmission chain.

In a first an access control list is used to assign a predefined or user-defined class to any data packet and thus classifies every data packet. Please note that predefined classes, e.g. local-voice, local-default and default, accrue from the SmartNode itself and therefore do not need any packet classification. Next the link arbiter handles each class corresponding to a user-defined service-policy profile.

Therefore 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.

The following sections describe methods to assign and share bandwidth on an interface in more details.

Quick references

The following sections are useful for administrators familiar with Cisco's IOS QoS features and having to become acquainted with SmartWare QoS configuration will find a helpful command cross reference.

Setting the modem rate

To match the voice and data multiplexing to the capacity of the access link is, with SmartWare, just one of a variety of possible configurations that you can set.

1. Create a minimal profile.

```
profile service-policy modem-512
  rate-limit 512
  source class local-voice
    priority
  source class local-voice
    share 30
  source class default
    share 70
```

2. Apply the profile just created to the interface connected to the modem.

```
context ip
interface wan

  use profile service-policy modem-512 out
```

If enough bandwidth is available the configuration could be even reduced as following. This configuration assumes:

```
profile service-policy modem-512
  rate-limit 2048
  source class local-voice
    priority
  source class local-voice
    priority
  source class default
```

Command cross reference

Comparing SmartWare with the Cisco IOS QoS software command syntax often helps administrators to straightforwardly configure SmartNode devices. In table 8 the Cisco IOS Release 12.2 QoS commands are in contrast with the respective SmartWare commands.

Table 8. 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 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. Tasks marked with advanced are only for administrators with sophisticated knowledge.

- Defining the access control list profile
- Assigning bandwidth to traffic classes (see page 144)
- Creating a top-level service policy profile (see page 147)
- Specifying source classes or lower level source policy profiles (see page 150)
- Devoting the service policy profile to an interface (see page 156)
- Displaying link arbitration status (see page 157)
- Displaying link scheduling profile information (see page 157)
- Enable statistics gathering (see page 158)

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*. After a packet has been classified, the packet needs to be marked by setting designated bits in the IP header.

In SmartWare access control lists are used for packet classification. Therefore a specific service policy has to be defined in a first step, which is used afterwards in an access control list as an optional *class of service* (cos) group. Refer to chapter 17, "Access control list configuration" on page 193, for more details about access control lists.

Packet classification using access control lists means to define outgoing traffic that is permitted on an interface based on the type of protocol. In combination with the optional cos group, which specifies the priority for a certain traffic type, a very powerful packet classification method is available.

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.

The packet classification has to be defined as a first step. In SmartWare the access control list are used to tag packets to a certain class.

Note Keep in mind that the class local-default comprises all other classes, which are not explicitly stated in a service policy profile.

Creating an access control list

The procedure to create an access control list is described in detail in chapter 17, "Access control list configuration" on page 193.

At this point an only exemplary situation is described that shows the necessary steps to tag any outbound traffic from a Web server, since this is used for the scenario depicted in figure 31. In this scenario the web server is regarded as a single source host when defining the access control list. Therefore 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.

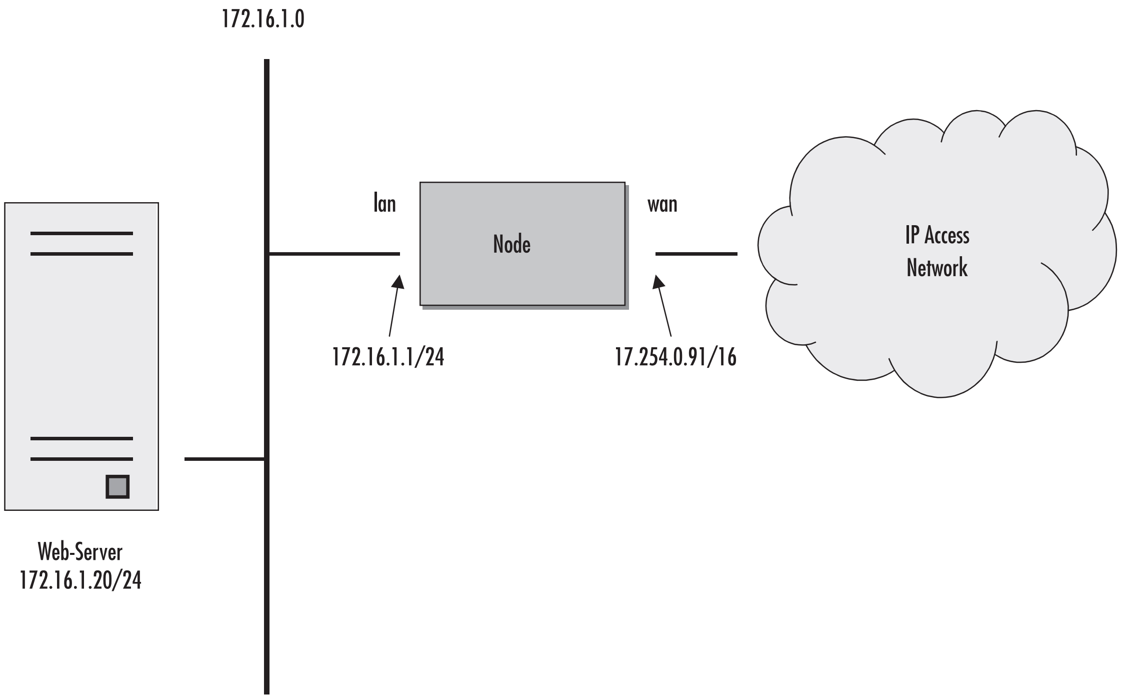


Figure 31. Scenario with Web server regarded as a single source host

A new access control list has to be created. Since the access control list is used for tagging Web server traffic, it gets the name *Webserver*. Moreover the IP filter of the access control list is used to define, that any packets matching the filter belong to a certain class of service. In our example the class of service that represents outbound Web related traffic is named *Web*.

You must pay attention to the fact, that access control list have an implicit “deny all” entry at the very end. Therefore if any packet that does not match the first criteria of outbound Web related traffic, would be dropped. This would be an improper behavior, because packets that are not related to outbound Web traffic need to be further processed in the link arbiter. As a result a second access control list entry is necessary that any other traffic is allowed.

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** *name*	Creates a new access control list profile named *name*
2	***node*(pf-acl)[*name*]#permit ip host ip-address any cos** *cos-name*	Creates an IP access control list entry that permits access for host at IP address *ip-address*, and specifies that packets matched by this rule belong to the class of service *cos-name*.
3	***node*(pf-acl)[*name*]#permit ip any** *any*	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 class of service *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)[Webserv~]#permit ip host 172.16.1.20 any cos Web
SN(pf-acl)[Webserv~]#permit ip any any
```

After packet classification is done using access control lists to tag packets to a certain class, as introduced in the preceding chapter, the link arbiter needs rules defining how to comply with all the traffic classes. For that purpose the network administrator creates a service policy profile. The service policy profile is in particular used to define how the link arbiter has to share the available bandwidth among several traffic classes on a certain interface.

The following sections describe methods to assign and share bandwidth on an IP interface.

Assigning bandwidth to traffic classes

To manage the access link bandwidth basically means to determine the order in which packets of the different classes are served.

Priority

One way of ordering packets is to give priority to one class and to serve the others 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. But we can only give the voice packets such an exclusive treatment because we know that they will not use up the whole bandwidth and that they will leave enough space for the lower priority classes.

Note The SmartWare link scheduler before any other bandwidth sharing method always serves the priority scheme first.

Weighted fair queuing

With other traffic sources you do not know as well, you would normally want to specify that for example class A gets three times the bandwidth of class B and that-no matter how many class A packets arrive-B packets will still be served. This is called weighted fair queuing: each class is assigned a *relative* weight, which determines how much more A bytes will be carried than B bytes. It would be possible to assign A the weight 3 and B the weight 1, but it is better using values that can be read as percentage, e.g. A is set to 75 and B is set to 25, which represents 75% for A and 25% for B of the whole bandwidth.

There is one thing to remember about weighted fair queuing (wfq), if in our example no class A packets are waiting in their queue-class B packets will get all the available bandwidth. In addition if three or more classes share the full bandwidth and one class is not active for a while, the remaining classes share the full bandwidth among each other-thereby getting a higher *relative* weight-if they are active at this time. The following example illustrates this behavior. Assume that relative weights are set, with A = 25%, B = 25%, and C = 50%. Under the circumstances, that C is not active for a while, A and B will each get half the bandwidth if they are both active at this moment.

Shaping

There is another commonly used way to assign bandwidth. It is called *shaping* and it makes sure that each class will get just as much bandwidth as assigned and not more. This is useful if some bandwidth is reserved for other applications in the network, which promises to deliver the packets within a guaranteed time. But in turn the network will insist that no more data is sent than reserved and paid for. When connecting the SmartNode to a *DiffServ* network an arbitration technique using shaping is recommended.

Burst tolerance

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 such that they may try their luck and send out bursts, which arrive with less delay when no one else uses the bandwidth. But this usage does not apply to access links.

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. SmartWare allows setting the burst rate and bursting size if more control over its behavior, which will rarely be the case, 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 *priority* to short data transfers. This feature is sometimes referred to as a *service curve*.

Hierarchy

So far you get introduced into different ways for assign bandwidth, you could think of it as different methods to decide whose packet have to be served next. Having more flexibility is available with SmartWare by cascading such bandwidth decisions one by the next. This is handled by means of setting up a decision tree where in each stage the branch to be served is chosen until at the level of a leaf the respective class is chosen. This procedure is called *hierarchical scheduling*.

The hierarchical notation has two advantages:

- It is a clean way of combining different *decision criteria*
- It can also be a nice way to cope with groups of traffic classes

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.

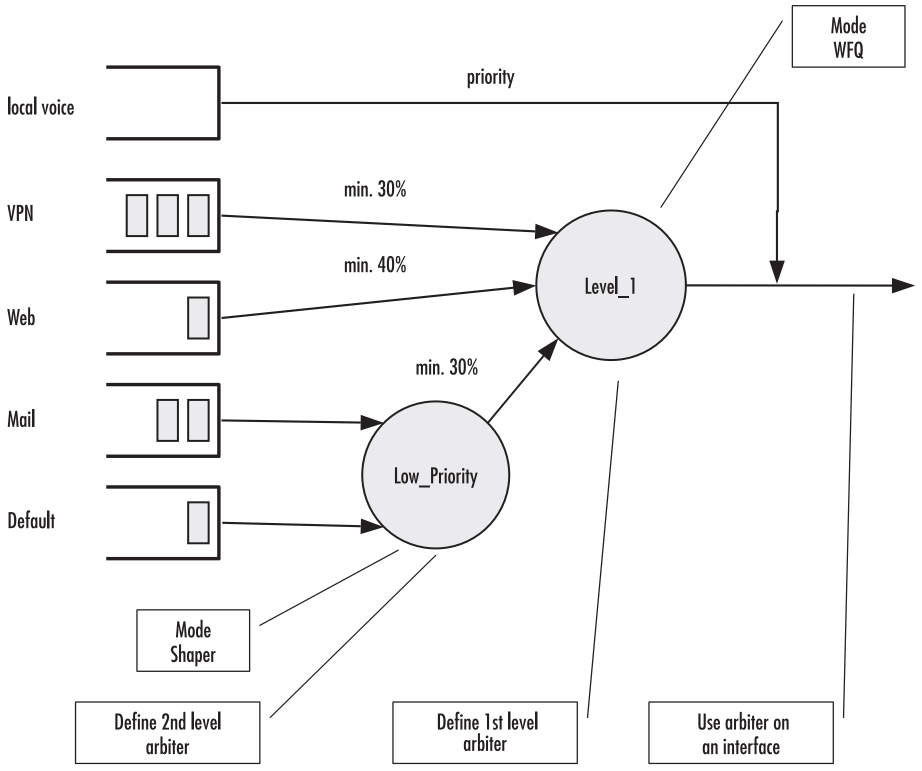


Figure 32. Example of Hierarchical Scheduling

Parts of the configuration used for arbiter *Level_1* are listed below. Take account of the line rate-limit on the top which is responsible for limiting the overall output to 512 kbps. The arbitration uses weighted fair queuing to share the bandwidth among the two classes VPN and Web and the policy Low_Priority. The source class local-voice has absolute priority and is therefore effectively bypassing the link arbiter, as shown in figure 33. The VPN class gets at least 30%, the Web class gets at least 40%, and the Low_Priority policy gets 30% of the overall bandwidth. Moreover there is defined that a maximum of 50 packets are queued for the source class Web.

```
profile service-policy Level_1
rate-limit 512
mode wfq
source class local-voice
priority
source class VPN
share 30

source class Web
share 40

source policy Low_Priority
share 30
```

Figure 33. Service policy for level_1 arbiter

Parts of the configuration used for arbiter *Low_Priority* are listed below. The arbitration uses shaping to share the bandwidth between the two classes Mail and Default. Therefore each class gets just as much bandwidth as assigned, but never more. The source class Mail gets 40% and Default gets 60% of the remaining bandwidth as defined for arbiter *Level_1*.

```
profile service-policy Low_Priority
mode shaper
source class mail
rate 40

source class default
rate 60
```

Figure 34. Service policy for low_priority arbiter

At last the *Level_1* service policy profile has to be used on the outbound IP interface. As shown in the example in figure 35 a possible related IP interface could be named as “wan”.

```
context ip
interface wan
use profile service-policy Level_1 out
```

Figure 35. IP Interface “wan” uses service policy profile

Creating a top-level service policy profile

Each service policy profile defines how the link arbiter has to act as. The link scheduler allows building up hierarchical scheduling. The uppermost service policy profile, also referred to as top-level service policy profile, is that service policy profile with which the link arbiter starts performing.

The top-level service policy profile is made up of following components:

- Arbitration schema definition, like weighted fair queuing or shaping

- Rate limit definition, that limits the overall output bandwidth on an IP interface
- Definitions of several source classes or policy, each defining the characteristics of handling a certain traffic type.

Note Be aware that a service policy profiles can include a lower level service policy profile, which is regarded as a sub-level profile as the example in Figure 0-3 shows. Not all service policy profile commands can be used to define a sub-level profile, e.g. the commands priority and rate-limit are only available when defining a top-level service policy profile. Refer to section "Specifying source classes or lower level source policy profiles" on page 150, for more details.

The syntactical structure of a typical top-level service policy profile is listed in figure 36. We will see over the syntax in more detail. To simplify matters the lines of the policy profile listing are numbered.

```
profile service-policy name
rate-limit value
mode mode
source class name

source class name

source class name

source policy name
```

Figure 36. Syntax of a Top-Level Service Policy Profile

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 has to be entered in kilobits per second, i.e. for 512 kbps the value 512 is used as argument for the rate-limit command. Keep in mind that the rate limit must be defined in a service policy profile.

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.

On lines 4 to 9 the source classes are defined. When using weighted fair queuing (wfq) each user-specified source class needs a value specifying its contribute to the overall bandwidth. For this purpose the share command is used, which divides up the full bandwidth among source classes and policies.

At a certain place the source class *default* has to be specified, e.g. in the top-level or a sub-level service policy profile configuration. This class is used, since it defines how packets, which do not belong to an explicit class and are not originating from the SmartNode itself, have to be processed.

On line 10 a source policy is included in the top-level service policy profile. The definitions for this source policy have to be done within another link arbiter profile. Within the top-level link arbiter policy the source policy—in this case the sub-level profile—only gets a relative share assigned.

This procedure describes creating a top-level service policy profile, which includes several source classes, or lower level service policy profiles.

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 {class | policy}** *src-name*	Enters source configuration mode for an ACL 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 source classes or lower level source policy profiles have to be entered.
6	***node (src)[src-name]exit***	Leaves the source configuration mode of a class or a lower level service policy profile.
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

Example: Creating a top-level service policy profile

The following example shows how to create a top-level service policy profile named *Sample*. This profile does not include any sub-level source policy profiles. The bandwidth of the outbound IP interface is limited to 512 kbps therefore 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 class local-voice
priority
source class Web
share 30
source class local-default
share 20
source class default
queue-limit 40
share 50
```

Note Keep in mind that the class *local-default* comprises all other classes originating from the SmartNode itself, which are not explicitly stated within the service policy profile.

Specifying source classes or lower level source policy profiles

Several commands are available to specify how source classes or lower level source policy profiles have to behave when transferring packet belonging to them.

Defining fair queuing weight

The command **share** is used with wfq link arbitration to assign the weight to the selected source class. When defining a number of source classes, the values are relative to each other. It is recommended to split 100—which is interpreted as 100%—among all available source classes, e.g. with 20, 30 and 50 as value for the respective share commands, which represent 20%, 30% and 50%.

This procedure describes defining fair queuing weight

Mode: Source

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

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 never more), when overloaded the shaper will behave like a wfq arbiter. Bit-rate specification for shaper (kilobits)

This procedure describes defining the bit-rate

Mode: Source

Command	Purpose
***node*(src)[name]#rate** [*kilobits* | **remaining]**	Defines the (average) bit-rate to the selected in kbps *kilobits* or as *remaining* if a second priority source is getting the unused bandwidth for the selected class or policy *name*

Defining absolute priority

This command **priority** can only be applied to classes, but not to lower level polices. The class is given absolute priority effectively bypassing the link arbiter. Care should be taken, as traffic of this class may block all other traffic. The packets given “priority” are taken into account by the “rate-limit”. Use the command *police* to control the amount of “priority” traffic.

Mode: Source

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

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.

This procedure describes defining maximum queue length

Mode: Source

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

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). Only one of these bits can be turned on. Each bit determines a specific method for the router to select a path.

Table 9. TOS values and their meaning

TOS Value	SmartWare Value	Meaning
1000	8	This bit minimizes the delay. The delay bit tells the link scheduler to choose high speed to minimize delay. This bit would typically be set for voice.
0100	4	This bit maximizes throughput. The throughput bit specifies high capacity links used for bulk transfers.
0010	2	This bit maximizes reliability. Routing protocols and network management applications use this for fault tolerant paths.
0001	1	This bit minimizes monetary costs. The cost bit is for low priority applications such as NNTP and signifies the lowest cost path.
0000	0	If all bits are cleared normal service is selected, which is referred as "default TOS".

Because TOS values are integers rather than sets of bits, computing the logical OR of two TOS values is not meaningful. For example, it would be a serious error for a router to choose a low delay path for a packet whose requested TOS was 1110 simply because the link scheduler noted that the former *delay bit* was set.

This procedure describes defining the type-of-service (TOS) field

Mode: Source

Command	Purpose
***node*(src)[*name*]#set ip tos** *value*	Defines the type-of-service (TOS) value applied to packets of for the selected class or policy *name*. Valid values for *value* are 0, 1, 2, 4, and 8, as given in table 9 on page 151.

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 values 6 and 7 are reserved for network control packets. Therefore, with the **set ip precedence** command the values 0 through 5 can be set for priority based on IP networks or applications.

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

This procedure describes defining the precedence field

Mode: Source

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

Specifying differentiated services codepoint marking

Differentiated services enhancements to the Internet protocol are intended to enable scalable service discrimination in an IP network without the need for per-flow state and signaling at every hop. A variety of services may be built from a small, well-defined set of building blocks, which are deployed in network nodes. The services may be either end-to-end or intra-domain; they include both those that can satisfy quantitative performance requirements (e.g., peak bandwidth) and those based on relative performance (e.g. "class" differentiation). Services can be constructed by a combination of:

- Setting bits in an IP header field at network boundaries (autonomous system boundaries, internal administrative boundaries, or hosts)
- Using those bits to determine how packets are forwarded by the nodes inside the network
- Conditioning the marked packets at network boundaries in accordance with the requirements or rules of each service.

The requirements or rules of each service must be set through administrative policy mechanisms, which are outside the scope of this user guide. A differentiated services-compliant network node includes a classifier that selects packets based on the value of the DS field, along with buffer management and packet scheduling mechanisms capable of delivering the specific packet forwarding treatment indicated by the DS field value. Setting of the DS field and conditioning of the temporal behavior of marked packets need only be performed at network boundaries and may vary in complexity.

In the IP header field, called the DS (for differentiated services) field is illustrated. Six bits of the DS field are used as a codepoint (DSCP) to select the per-hop behavior (PHB) a packet experiences at each node. A two-bit currently unused (CU) field is reserved and its definition and interpretation are outside the scope of this document. Differentiated services-compliant nodes when determining the per-hop behavior to apply to a received packet ignore the value of the CU bits.

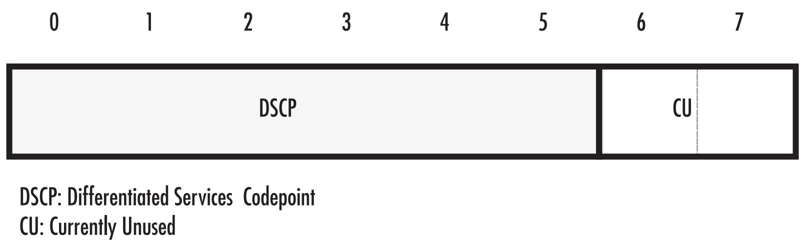


Figure 37. DS field structure

In a DSCP value notation 'xxxxxx' (where 'x' may equal '0' or '1') used in this user guide, the left-most bit signifies bit 0 of the DS field (as shown in figure 37), and the right-most bit signifies bit 5.

The DSCP field within the DS field is capable of conveying 64 distinct codepoints. The codepoint space is divided into three pools for the purpose of codepoint assignment and management: a pool of 32 recommended codepoints (Pool 1) to be assigned by Standards Action, a pool of 16 codepoints (Pool 2) to be reserved for experimental or local use (EXP/LU), and a pool of 16 codepoints (Pool 3) which are initially available for experimental or local use, but which should be preferentially utilized for standardized assignments if Pool 1 is ever exhausted. The pools are defined in the table 10 (where 'x' refers to either '0' or '1'):

Table 10. Codepoint pools

Pool	Codepoint Space	Assignment Policy
1	xxxxx0	Standards Action
2	xxxx11	EXP/LU
3	xxxx01	EXP/LU (*)

(*) may be utilized for future Standards Action allocations as necessary

Service providers are not required to use the same node mechanisms or configurations to enable service differentiation within their networks, and are free to configure the node parameters in whatever way that is appropriate for their service offerings and traffic engineering objectives. Over time certain common per-hop behaviors are likely to evolve (i.e. ones that are particularly useful for implementing end-to-end services) and these may be associated with particular EXP/LU PHB codepoints in the DS field, allowing use across domain boundaries. These PHBs are candidates for future standardization.

Note If you have to configure service differentiation on your SmartNode, ensure to that codepoint settings are arranged with your service provider.

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

This procedure describes defining the codepoints in the DS field

Mode: Source

Command	Purpose
***node*(src)[*name*]#set ip dscp** *value*	Defines the Differentiated Services Codepoint value applied to packets of for the selected class or policy *name*. The range for *value* 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 11 lists the tag components.

Table 11. 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.

This procedure describes defining the 3-bit priority field (802.1p)

Mode: Source

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

Defining random early detection

The command **random-detect** is used random early detection (RED) for queues that carry lots of TCP transfers that last longer than simple web requests. In such a case there is a risk that TCP flow-control might be ineffective. With RED configured the queue occasionally drops packets when the queue has grown over half the “queue-limit”. This improves the performance of TCP flow-control. 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.

This procedure describes defining random early detection

Mode: Source

Command	Purpose
***node*(src)[*name*]#random-detect {*burst-tolerance*}**	Defines random early detection (RED) for queues of for the selected class or policy *name*. The range for the optional value *burst-tolerance* 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** *average-kilobits* **burst-size** *kilobits-ahead*	Defines how traffic arriving in a queue for the selected class or policy *name* has to be controlled. The value *average-kilobits* for average rate permitted is in the range from 0 to 10000 kbps. The value *kilobits-ahead* 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 38.

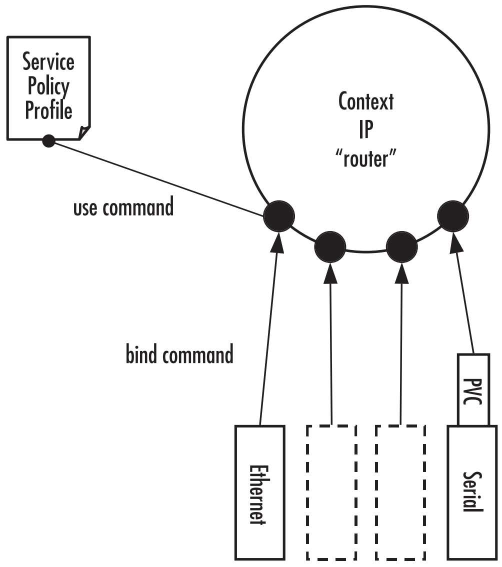


Figure 38. Using a Service Policy Profile on an IP Interface

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. The command offers an optional argument allowing to define that the service policy profile is activated in receive or transmit direction.

Note Be aware that service policy profiles can only be activated on the transmit direction at the moment!

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.

This procedure describes devoting a service policy profile to an IP interface that is defined within the IP context

Mode: Interface

Step	Command	Purpose
1	***node***(**if-ip**)[*if-name*]**#use profile service-policy** *name* **{in | out}**	Applies the service policy profile *name* to the selected interface *if-name*. 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.

This procedure describes displaying the link arbitration status on all interfaces or a certain interface.

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.

This procedure describes displaying link scheduling profile information of an existing service-policy profile

Mode: Administrator execution

Step	Command	Purpose
1	***node*#show profile service-policy** *name*	Displays link scheduling profile information of the service-policy profile *name*

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 value, which defines the level of detail. The value is in the range from 1 to 6. In table 12 below the values and their implications are compiled.

Table 12. Values defining verbosity of command output

Optional Value	Implication on Command Output
0	Statistic gathering is switched off
1	Display amount of packets transferred
2	
3	
4	
5	
6	

Note The debug features offered by SmartWare require 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.

This procedure describes enabling the statistic gathering of link scheduler operations with a certain verbosity of the command output.

Mode: Source

Step	Command	Purpose
1	***node***(**src**)[*name*]**#debug queue statistics** *level*	Enables statistic gathering for the selected class or policy *name*. The optional argument *level*, which is in the range from 1 to 6, defines the verbosity of the command output.

Example: Enable statistics gathering

The following example shows how to enable statistic gathering for the source class Web, which is defined in the service policy profile *Level_1*. The level of verbosity for the command output is set to 6, which is very verbose.

```
SN>enable
SN#configure
SN(cfg)#profile service-policy Level_1
SN(pf-srvpl)[Level_1]#source class Web
SN(src)[Web]#debug queue statistics 6
```

Chapter 14 **Serial port configuration**

Chapter contents

Introduction

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

- Serial port configuration task List
- Configuration tasks
- Examples

The SmartNode 2000 series device supports both 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 SmartNode device for specific information regarding the connector pinout and the selection of cables to connect with 3rd party equipment.

The SmartNode 2000 series 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 then are 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 161)
- Enabling an interface (see page 161)
- Configuring the serial encapsulation type (see page 162)
- Configuring the hardware port protocol (see page 163)
- Configuration of the active clock edge (see page 164)
- Entering Frame Relay mode (see page 165)
- Configuring the LMI type (see page 165)
- Configuring the keep-alive interval (see page 166)
- Enabling fragmentation (see page 167)
- Entering Frame Relay PVC configuration mode (see page 168)
- Configuring the PVC encapsulation type (see page 169)
- Binding the Frame Relay PVC to IP interface (see page 169)
- Disabling a Frame Relay PVC (see page 170)
- Displaying Frame Relay information (see page 171)

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 is to prevent 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 desribes 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 2300 series device. Check that in the command output of **show port serial** *State* is set to *CLOSED*.

```
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 displays.

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

This procedure desribes how to enable 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***]#no shutdown**	Enable 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 2300 series device. Check that in the command output of **show port serial** *State* is set to *OPENED*.

```
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 desribes how to set the encapsulation type of the serial interface for frame relay

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***]#encapsulation framerelay**	Sets the encapsulation for selected interface to frame relay
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 2300 series device. 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

This procedure desribes how to set the encapsulation type of the serial interface for frame relay

Mode: Administrator execution

Step	Command	Purpose
1	***node*(cfg)#port serial slo***t po**rt**	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 2300 series device. Check that in the command output of **show port serial** *Hardware Port* is set to *X.21*.

```
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 data is transmitted by the SmartNode.

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

Mode: Port serial

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

Example: Configuring the active clock edge

The following example enables sending of the data on the negative edge on slot 0 and port 0 of a SmartNode 2300 series device. Check that in the command output of **show port serial** *Transmit Clock* is set to *inverted*.

```
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 the tasks for configuring Frame Relay for the serial interface on a SmartNode series 2000 device, after setting the basic serial interface parameters according to the previous sections.

This procedure desribes how to enter the Frame Relay configuration mode

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***]#framerelay**	Enters the Frame Relay configuration mode
3	***node*(frm-rel)[***slot/port***]#**	Frame Relay configuration mode prompt is displayed

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 2300 series device.

```
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 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 desribes how to set the LMI type

Mode: Frame Relay

Step	Command	Purpose
1	***node*(frm-rel)[*slot/port*]#lmi-type {ansi | gof | itu}**	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 of a SmartNode 2300 series device.

```
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 desribes how to set the keep-alive interval

Mode: Frame Relay

Step	Command	Purpose
1	***node*(frm-rel)[*slot/port*]#keepalive** *number*	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 2300 series device.

```
SN(cfg)#port serial 0 0
SN(prt-ser)[0/0]#framerelay
SN(frm-rel)[0/0]#keepalive 10
```

Enabling fragmentation

SmartWare supports 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 kps). As opposed to IP fragmentation (also supported by SmartWare) Frame Relay fragmentation is transparent to the IP layer thus leaving IP packets unchanged which may be important for IP based applications susceptible to IP fragmentation.

This procedure desribes how to enable Frame Relay fragmentation

Mode: Frame Relay

Step	Command	Purpose
1	***node*(cfg)#port serial** *slot port*	Selects the serial interface on slot and port.
2	***node*(prt-ser)[*0/0*]#framerelay**	Enter Frame Relay configuration mode.
3	***node*(frm-rel)[*0/0*]#use profile service-policy** *name* **out**	Use the previously defined service policy profile on Frame Relay layer (and not on IP interface level) in outward direction. For other profile settings refer to the software configuration guide.
4	***node*(frm-rel)[*0/0*]#fragment** *size*	Define the maximum size (in Bytes) of the Frame Relay payload (excluding Frame Relay header and trailer overhead) for all PVCs (FRF.12 end-to-end fragmentation). See also table below
5	***node*(frm-rel)[*0/0*]#pvc** *dlci*	Enters the PVC configuration mode by assigning a DLCI number to be used on the specified virtual circuit.
6	***node*(pvc)[*dlci*]#fragment** *size*	Define the maximum size (in Bytes) of the Frame Relay payload (excluding Frame Relay header and trailer overhead) for this PVC only (FRF.12 interface fragmentation). See also 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. Also do not use the frame relay service policy profile on IP layer but rather on 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 and 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 that are 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 2300 series device is done in the PVC configuration mode. The DLCI has to be in the range from 1 to 1022.

Note There is a maximum of eight PVCs that can be defined.

This procedure desribes how to enter the PVC configuration

Mode: Frame Relay

Step	Command	Purpose
1	***node*(frm-rel)[*slot/port*]#pvc** *dlci*	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 2300 series device.

```
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

To set the encapsulation type to comply with the Internet Engineering Task Force (IETF) standard (RFC 1490) the PVC configuration command **encapsulation rfc1490** has to be used. Use this keyword when connecting to another vendor's equipment across a Frame Relay network.

This procedure desribes how to set the encapsulation type to comply with RFC 1490

Mode: Frame Relay

Step	Command	Purpose
1	***node*(frm-rel)[*slot/port*]#encapsulation rfc1490**	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 2300 series device.

```
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 defined already 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 39 below.

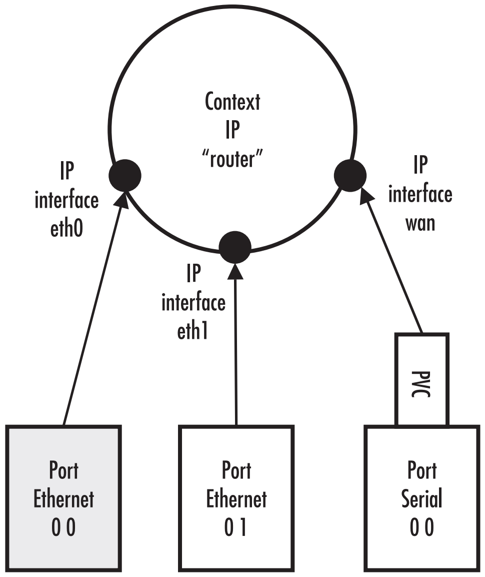


Figure 39. IP interface 'wan' is bound to PVC 1 on port serial 0 0

This procedure desribes 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	***node***(**pvc)**[*dlci*]**#bind interface** *name* **router**	Binds Frame Relay PVC dlci 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 2300 series device.

```
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
```

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 desribes how to disable the Frame Relay PVC dlci on the serial interface

Mode: PVC

Step	Command	Purpose
1	***node*(pvc)[dlci]#shutdown**	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 2300 series device.

```
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 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:
#----------------------------------------------------------------#
#                                                                #
# SN2300                                                         #

pvc 1
  encapsulation rfc1490
  bind interface wan router
  shutdown
  exit
```

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 desribes 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 displays the Frame Relay configuration settings for the serial interface.

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

Framerelay Configuration:
Port            LMI-Type       Keepalive      Fragmentation
-----------------------------------------------------------
serial 0 0 0    ansi           10             enabled

PVC Configuration:
Port            DLCI      State     Encaps    Binding
-----------------------------------------------------
serial 0 0 0    1         open      rfc1490   wan@router
```

Examples

Displaying serial port information

The following example shows the commands used to display serial port configuration settings for a SmartNode 2300 series device. Moreover a typical command output is listed below.

```
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

The following example shows the commands used to display Frame Relay configuration settings for a Smart-Node 2300 series device. Moreover a typical command output is listed below.

```
SN>enable
SN#configure
SN(cfg)#show framerelay

Framerelay Configuration:
Port           LMI-Type       Keepalive      Fragmentation
----------------------------------------------------------
serial 0 0 0   ansi           10             enabled

PVC Configuration:
Port           DLCI      State     Encaps    Binding
----------------------------------------------------
serial 0 0 0   1         open      rfc1490   wan@router
```

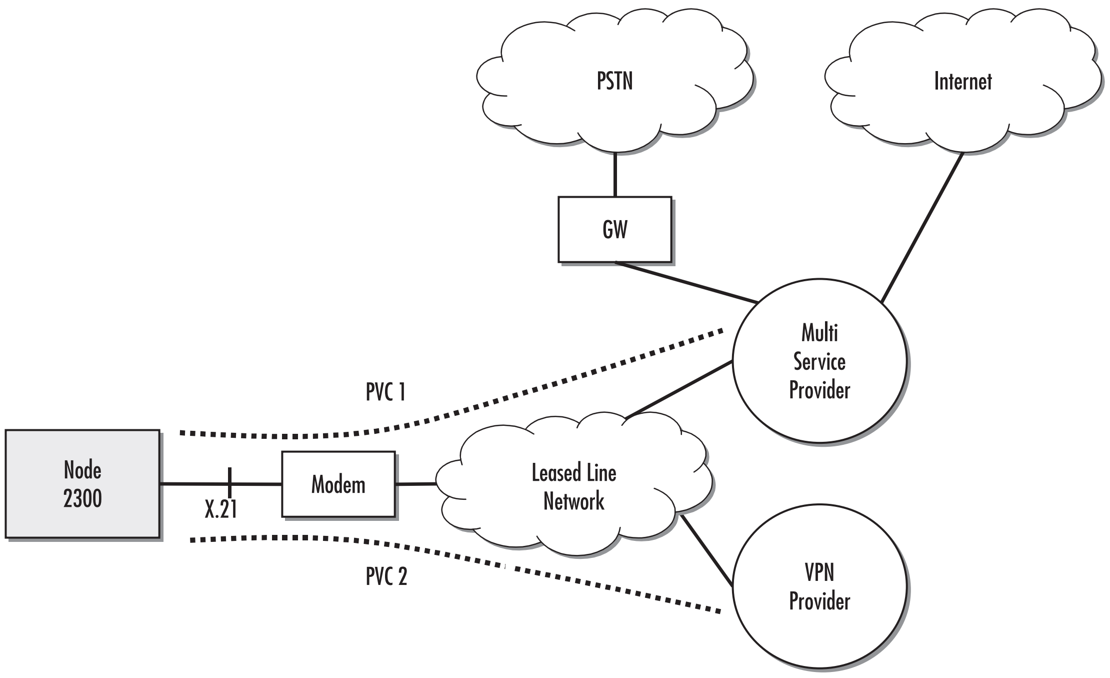


Figure 40. Typical Integrated Service Access Scenario with dedicated PVCs

Integrated service access

The example in figure 40 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 2300 series device.

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 are network based on IP. Since both providers are working independently the SmartNode 2300 series device 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 2300 series device is working as a DTE and accesses the leased line network via a leased line modem connected to the serial interface. On the serial interface on slot 0 and port 0 the hardware port protocol X.21 is used.

Devices accessing the MSP and VPN services are attached to the 100 Mbps Ethernet port 0/0 on the SmartNode 2300 series device. 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 40 has to be configured for the SmartNode 2300 series device. 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 2300 series device 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.

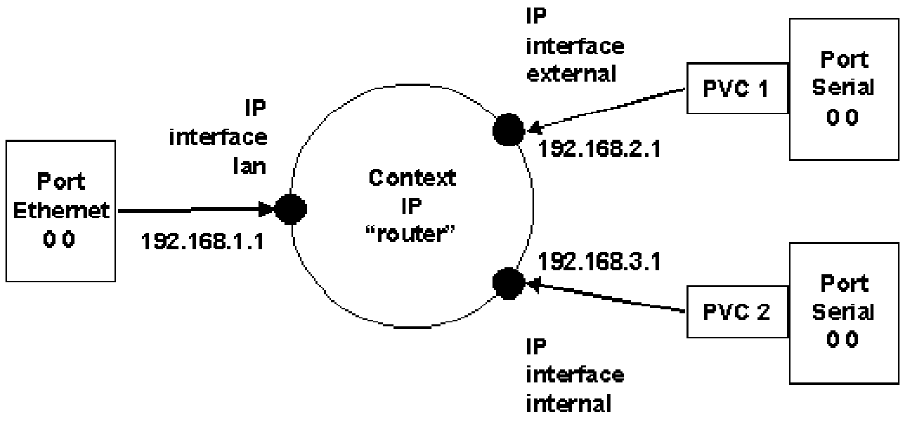


Figure 41. IP Context with logical IP interfaces bound to Ethernet port, serial port PVC 1 and PVC 2

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. The IP interface configuration is set up 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. We define a voice profile which gives priority to voice packets. We set the rate limit according to the bandwidth available for voice and data on PVC 1 (512kBits/sec 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. The serial interface settings are configured.

```
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. 5

The Frame Relay configuration is done next. Therefore we have to change to the Frame Relay configuration mode. We 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. The introduced PVCs need to be configured. We enable fragmentation for PVC 1. The voice uses codec G.723 at a packet size of 30ms. Therefore 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 to 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]#pvc 2
SN(pvc)[2]#encapsulation rfc1490
SN(pvc)[2]#bind interface internal router
```

7. Finally we 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             300


PVC Configuration:
Port           DLCI     State     Fragment  Encaps    Binding
--------------------------------------------------------------
serial 0 0 0   1        close     disabled  rfc1490   external@router
serial 0 0 0   2        close     disabled  rfc1490   internal@router
```

Chapter 15 Basic IP routing configuration

Chapter contents

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 — that is, 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 179)
- Displaying IP route information (see page 180)

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	***node***(**cfg)#context ip router**	Enters the IP router context
2	***node***(**ctx-ip)[router]#route** *network mask* {*address* | *interface*} [*metric*]	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 device at 172.17.100.2. Ethernet port 0 1 has the address 172.17.100.1/24 and is bound to 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 network via 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             local        1    UL           0
* 20.0.0.0/24        172.17.100.2    static       0    U            0
* 172.16.0.0/16      lan             local        1    UL           6
```

Examples

Basic static IP routing example

Figure 42 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*.

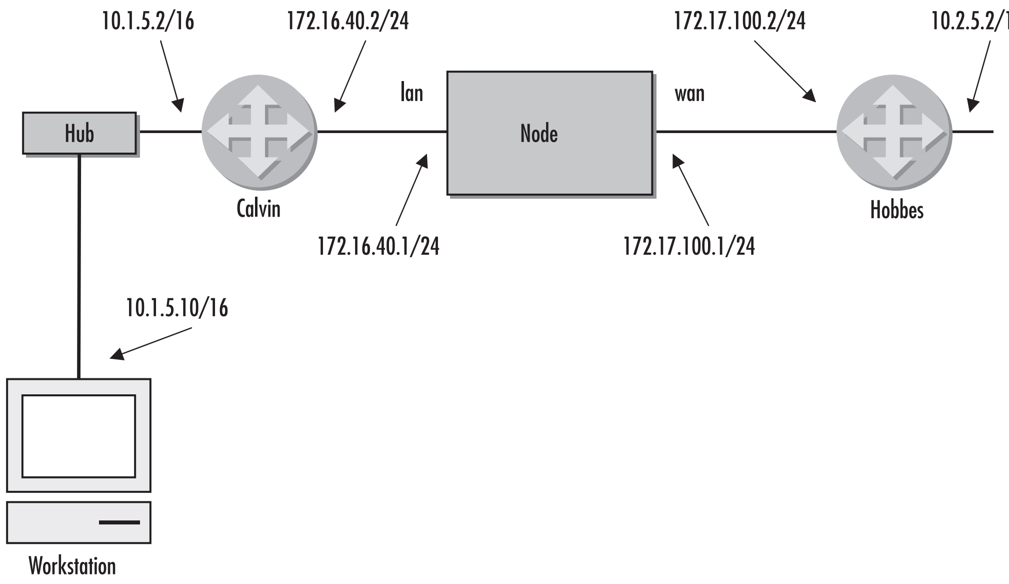


Figure 42. Internetwork with three routers and four networks

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        0    LHG        n/a
* 172.16.40.1/24                      local        0    LHG        n/a
* 172.17.100.1/24                     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 16 RIP configuration

Chapter contents

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 185)

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.

The 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 186)
- Specifying the send RIP version (see page 186)
- Specifying the receive RIP version (see page 187)
- Enabling RIP learning (see page 187)
- Enabling an interface to receive RIP (see page 188)
- Enabling RIP announcing (see page 188)
- Enabling RIP auto summarization (see page 189)
- Specifying the default route metric (see page 189)
- Enabling RIP split-horizon processing (see page 190)
- Enabling the poison reverse algorithm (see page 190)
- Enabling holding down aged routes (see page 191)
- Displaying RIP Configuration of an IP interface (see page 191)
- Displaying global RIP information (see page 192)

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	***node*(if-ip)[*name*]#rip supply**	Enables send RIP on interface *name*

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 *name*

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, the SmartWare application software sends RIP 1compatible 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 *name*

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, the SmartWare application software receives RIP version 1 and version 2 packets. The 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	***node*(if-ip)[*name*]# rip receive version {1 | 1or2 | 2}**	Selects receive RIP version for interface *name*

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 exists. 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	***node*(if-ip)[*name*]# rip learn host**	Enables accepting of IP host routes received on interface *name*
2	***node*(if-ip)[*name*]#rip learn default**	Enables learning using a default route advertised by an RIP neighbor on interface *name*

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	***node*(if-ip)[*name*]#rip listen**	Enables receive RIP on IP interface *name*

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	***node*(if-ip)[*name*]#rip announce {default | host | self-as-default | static}**	Selects the RIP announcing method on interface *name*

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	***node*(if-ip)[*name*]#rip auto-summary**	Enables RIP auto-summarization on interface *name*

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 (that is, 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 *value* indicating the distance to the destination on interface *name*

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 *name*

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 *name*

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 *name*

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 *name*

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 17 Access control list configuration

Chapter contents

Introduction

This chapter provides an overview of IP Access Control Lists and describes the tasks involved in configuring them through the Patton SmartWare.

This chapter includes the following sections:

- About access control lists
- Access control list configuration task list (see page 196)
- Examples (see page 206)

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 that is focused on 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 43 host A is allowed to access the Human Resources network and host B is prevented from accessing the Human Resources network.

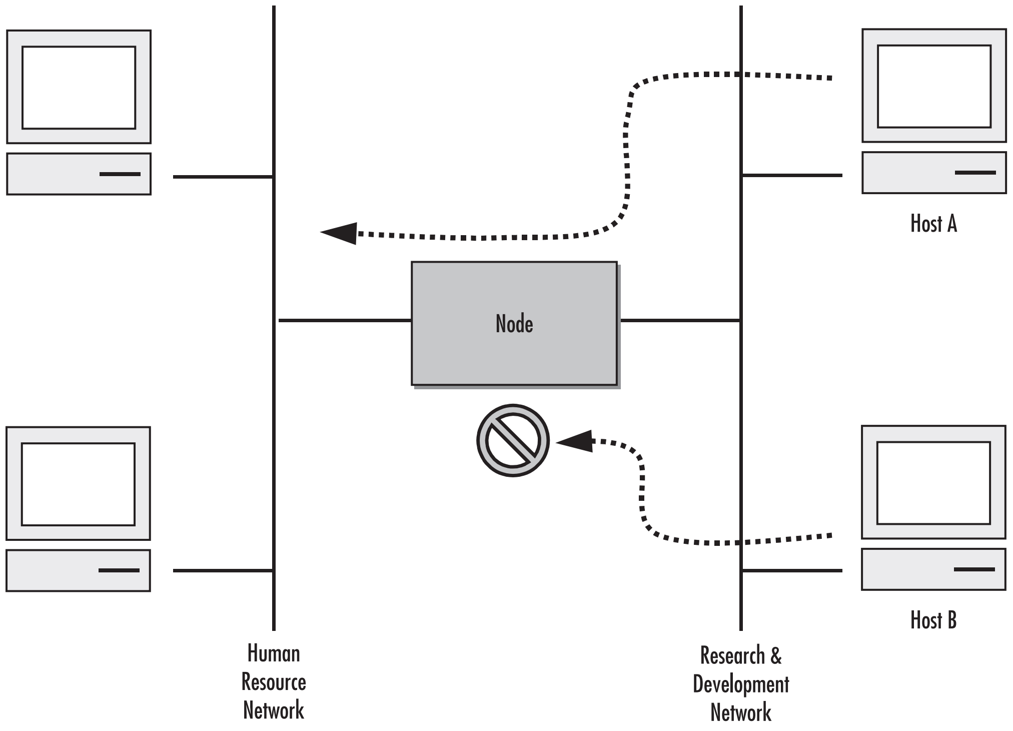


Figure 43. Using traffic filters to prevent traffic from being routed to a network

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.

- Map out the goals of the access control list
- Create an access control list profile and enter configuration mode (see page 197)
- Add a filter rule to the current access control list profile (see page 197)
- Add an ICMP filter rule to the current access control list profile (see page 199)
- Add a TCP, UDP or SCTP filter rule to the current access control list profile (see page 201)
- Bind and unbind an access control list profile to an IP interface (see page 203)
- Display an access control list profile (see page 204)
- Debug an access control list profile (see page 204)

Map 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, editing those entries online can be uncomfortable. Therefore we recommend editing multifaceted access control list offline within a configuration file and downloading the configuration file later via TFTP to your SmartNode device.

Create 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** *name*	Creates the access control list profile *name* 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]#
```

Add 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** {*src src-wildcard* **| any | host** *src*} {*dest dest-wildcard* **| any | host** *dest*} **[cos** *group*]	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 *src-wildcard*
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 "Quality of Service Configuration" later in this guide.
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)#
```

Add 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 *src-wildcard*.
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 "Quality of Service Configuration" later in this guide.
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)#
```

Add 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-wild-card* **| 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 an 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 *src-wildcard*.
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 "Quality of Service Configuration" later in this guide.
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)#
```

Bind and unbind 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 *if-name*

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 *if-name*

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.

Display 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** *name*	Displays the access control list profile *name*

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
```

Debug 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 *if-name* 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 *if-name*

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

Deny a specific subnet

Figure 44 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. Therefore 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*.

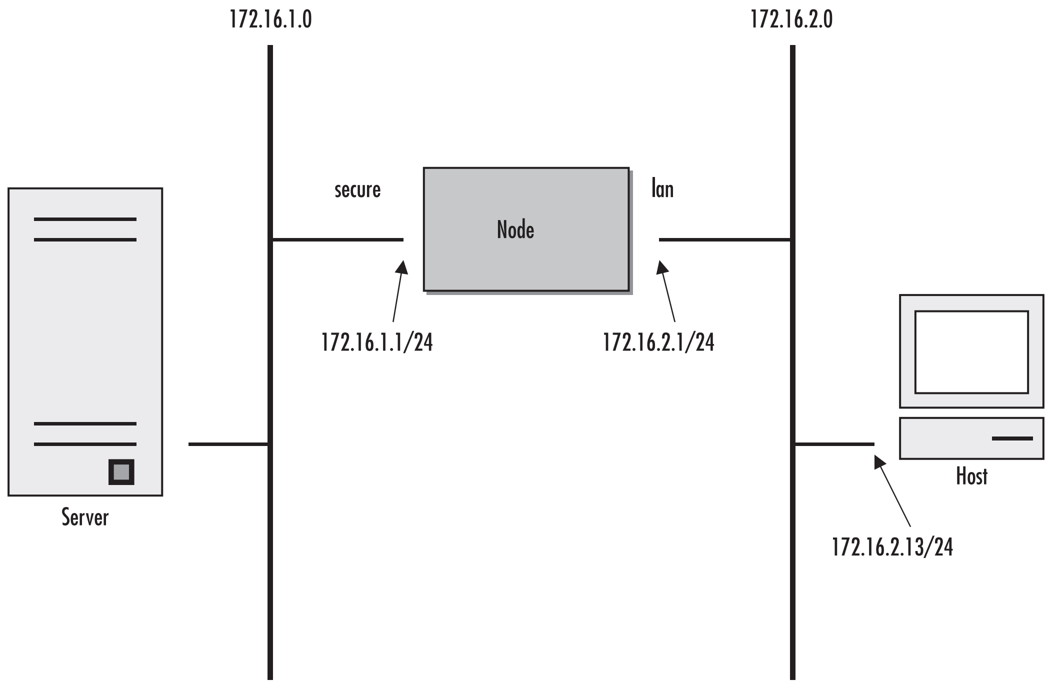


Figure 44. Deny a specific subnet on an interface

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 18 SNMP Configuration

Chapter contents

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 the Patton SmartWare.

This chapter includes the following sections:

- Simple Network Management Protocol (SNMP)
- SNMP tools (see page 211)
- SNMP configuration task list (see page 211)
- Using the AdventNet SNMP utilities (see page 217)
- Standard SNMP version 1 traps (see page 220)
- Patton enterprise-specific traps (see TBD)
- Patton private MIBs (see TBD)

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 the 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 13. The returned value when reading the sysObjectID object for each SmartNode model

SmartNode 4112	.iso.org.dod.internet.private.enterprises.inalp.products.sn4000.sn4112 1.3.6.1.4.1. 5349.2.4.5.5
SmartNode 4114	.iso.org.dod.internet.private.enterprises.inalp.products.sn4000.sn4114 1.3.6.1.4.1. 5349.2.4.5.6
SmartNode 4116	.iso.org.dod.internet.private.enterprises.inalp.products.sn4000.sn4116 1.3.6.1.4.1. 5349.2.4.5.7
SmartNode 4118	.iso.org.dod.internet.private.enterprises.inalp.products.sn4000.sn4118 1.3.6.1.4.1. 5349.2.4.5.8
SmartNode 4522	.iso.org.dod.internet.private.enterprises.inalp.products.sn4000.sn4522 1.3.6.1.4.1. 5349.2.4.5.2
SmartNode 4524	.iso.org.dod.internet.private.enterprises.inalp.products.sn4000.sn4524 1.3.6.1.4.1. 5349.2.4.5.1
SmartNode 4526	.iso.org.dod.internet.private.enterprises.inalp.products.sn4000.sn4526 1.3.6.1.4.1. 5349.2.4.5.3
SmartNode 4528	.iso.org.dod.internet.private.enterprises.inalp.products.sn4000.sn4528 1.3.6.1.4.1. 5349.2.4.5.4

Table 14. SmartNode Models and their Unique sysObjectID

SmartNode Model	SysObjectID
SmartNode 1200	.iso.org.dod.internet.private.enterprises.inalp.products.sn1200 1.3.6.1.4.1. 5349.2.4.1
SmartNode 1400	.iso.org.dod.internet.private.enterprises.inalp.products.sn1400 1.3.6.1.4.1. 5349.2.4.2
SmartNode 2300	.iso.org.dod.internet.private.enterprises.inalp.products.sn2300 1.3.6.1.4.1. 5349.2.4.3
SmartNode 2400	.iso.org.dod.internet.private.enterprises.inalp.products.sn2400 1.3.6.1.4.1. 5349.2.4.4
SmartNode 4112	.iso.org.dod.internet.private.enterprises.inalp.products.sn4000.sn4112 1.3.6.1.4.1. 5349.2.4.5.5

Table 14. SmartNode Models and their Unique sysObjectID (Continued)

SmartNode Model	SysObjectID
SmartNode 4114	.iso.org.dod.internet.private.enterprises.inalp.products.sn4000.sn4114 1.3.6.1.4.1. 5349.2.4.5.6
SmartNode 4116	.iso.org.dod.internet.private.enterprises.inalp.products.sn4000.sn4116 1.3.6.1.4.1. 5349.2.4.5.7
SmartNode 4118	.iso.org.dod.internet.private.enterprises.inalp.products.sn4000.sn4118 1.3.6.1.4.1. 5349.2.4.5.8
SmartNode 4522	.iso.org.dod.internet.private.enterprises.inalp.products.sn4000.sn4522 1.3.6.1.4.1. 5349.2.4.5.2
SmartNode 4524	.iso.org.dod.internet.private.enterprises.inalp.products.sn4000.sn4524 1.3.6.1.4.1. 5349.2.4.5.1
SmartNode 4526	.iso.org.dod.internet.private.enterprises.inalp.products.sn4000.sn4526 1.3.6.1.4.1. 5349.2.4.5.3
SmartNode 4528	.iso.org.dod.internet.private.enterprises.inalp.products.sn4000.sn4528 1.3.6.1.4.1. 5349.2.4.5.4

According to table 14, 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.

IMPORTANT

The SNMP agent running in the SmartWare is SNMP version 1 (SNMPv1) compliant. Therefore both SNMP version 2 (SNMPv2) and SNMP version 3 (SNMPv3) are not 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 the section "Using the AdventNet SNMP Utilities" 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 212)
- Setting access community information (required) (see page 214)
- Setting allowed host information (required) (see page 215)
- Specifying the default SNMP trap target (optional) (see page 215)

- Displaying SNMP related information (optional) (see page 216)

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** *name*	Sets the contact persons name
2	***node***(cfg)#**system location** *location*	Sets the system location
3	***node***(cfg)#**system hostname** *hostname*	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.

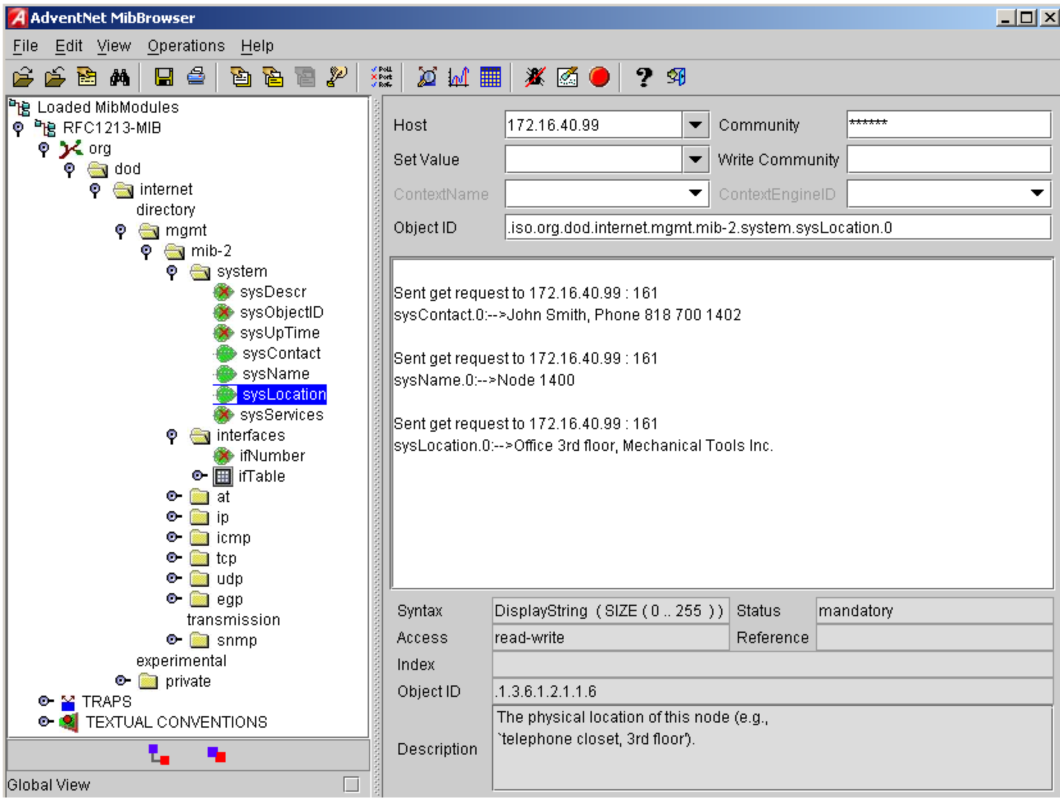
The MIB-II system group values are accessible for reading and writing via the following SNMP objects:

- .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 45.

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.



Host 172.16.40.99
Community ******
Set Value
Write Community
ContextName
ContextEngineID
Object ID .iso.org.dod.internet.mgmt.mib-2.system.sysLocation.0

```
Sent get request to 172.16.40.99 : 161
sysContact.0:-->John Smith, Phone 818 700 1402

Sent get request to 172.16.40.99 : 161
sysName.0:-->Node 1400

Sent get request to 172.16.40.99 : 161
sysLocation.0:-->Office 3rd floor, Mechanical Tools Inc.
```

Syntax	DisplayString (SIZE (0 .. 255))	Status	mandatory
Access	read-write	Reference	
Index			
Object ID	.1.3.6.1.2.1.1.6		
Description	The physical location of this node (e.g., `telephone closet, 3rd floor').		

Figure 45. 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 0-1 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 makes use of 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 Knowledge of read-only community strings allows read access to information that is stored on an affected device, leading to a failure of confidentiality. Knowledge of read-write community strings allows remote configuration of affected devices without authorization, possibly without the awareness of the administrators of the device and resulting in a failure of integrity and a possible failure of availability. Therefore defining a community strings which 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 password. Do not use easily guessable 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	***node*(cfg)#snmp community** *name* **{ ro I rw }**	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	***node***(**cfg**)**#snmp host** *IP-address-of-node* **security-name** *community*	Configures a host that with IP address *IP-address-of-node* can access the MIB of this SmartNode 1000, 2000 or 4000 series device, using the security name *community*.

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 Smart-Node 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** *IP-address-of-node* **security-name** *community*	Configures a SNMP trap target with IP address *IP-address-of-node* that receives trap messages of this SmartNode 1000, 2000 or 4000 series device, using the security name *community* 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 46 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 by 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 46.

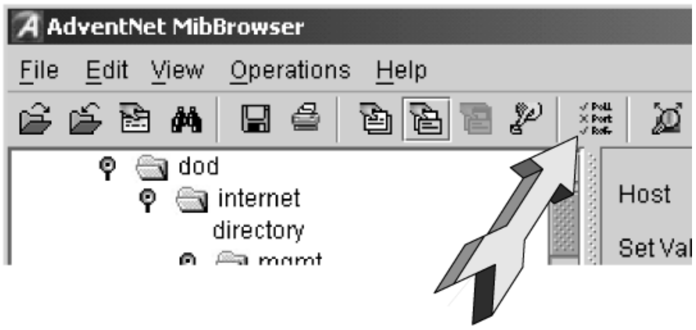


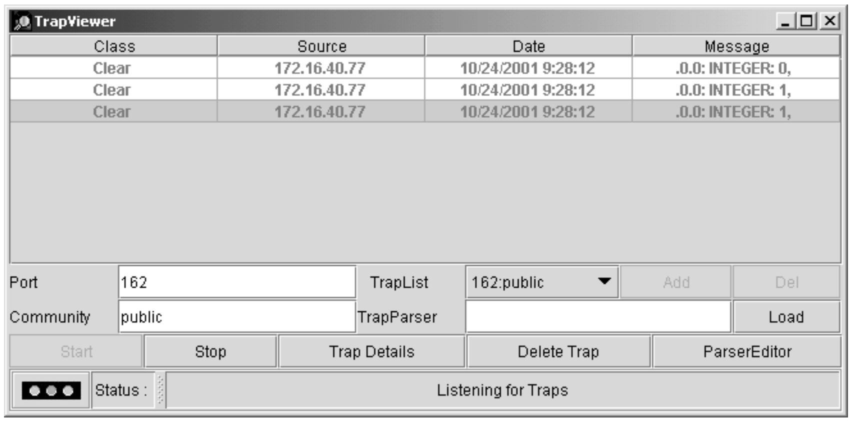
Figure 46. AdventNet MibBrowser Settings Button on the Toolbar

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 Smart-Node 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 217. Figure 47 is a screen shot of the TrapViewer.



Class	Source	Date	Message
Clear	172.16.40.77	10/24/2001 9:28:12	.0.0: INTEGER: 0,
Clear	172.16.40.77	10/24/2001 9:28:12	.0.0: INTEGER: 1,
Clear	172.16.40.77	10/24/2001 9:28:12	.0.0: INTEGER: 1,

Port: 162
TrapList: 162:public
Community: public
TrapParser:
Status : Listening for Traps

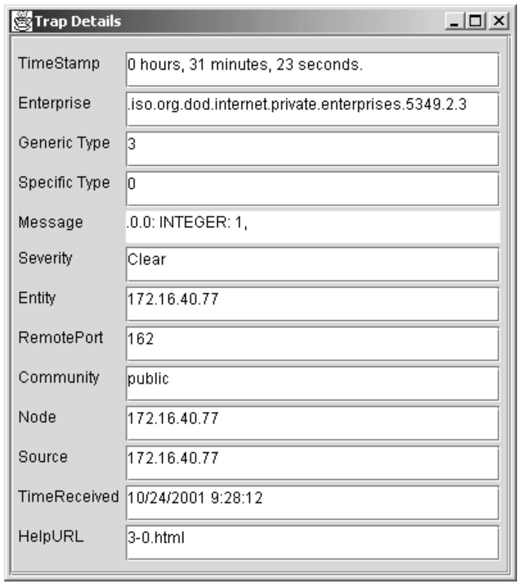
Figure 47. AdventNet TrapViewer displaying received traps

The TrapViewer has a table that depicts 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. On clicking this TrapViewer begins to receive traps as on specified port and community.
- The traps when received are depicted in the table in the TrapViewer. The trap table by default has the following four columns:
  - *Class* that defines the severity of the trap.
  - *Source* that depicts 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 48 show the screen of such a trap details window.



TimeStamp	0 hours, 31 minutes, 23 seconds.
Enterprise	.iso.org.dod.internet.private.enterprises.5349.2.3
Generic Type	3
Specific Type	0
Message	.0.0: INTEGER: 1,
Severity	Clear
Entity	172.16.40.77
RemotePort	162
Community	public
Node	172.16.40.77
Source	172.16.40.77
TimeReceived	10/24/2001 9:28:12
HelpURL	3-0.html

Figure 48. AdventNet Trap Details window of TrapViewer

The various details available in the Trap Details window are listed in table 15:

Table 15. Details available in the Trap Details window

Trap Details	Description
TimeStamp	The TimeStamp is a 32-bit unsigned value indicating the number of cent seconds that have elapsed since the (re)start of the SNMP agent and the sending of the trap. This field shows the value stored in the MIB-II sysUpTime variable converted into hours, minutes and seconds.
Enterprise	This field shows the OID of the management enterprise that defines the trap message. The value is represented as an OBJECT IDENTIFIER value and has a variable length.
Generic Type	The Generic type value is categorized and numbered 0 to 6. They are 0-coldStart, 1-warmStart, 2-linkDown, 3-linkUp, 4-authenticationFailure, 5-egpNeighborLoss. The trap type value 6 is identified as enterprise-specific value. This field shows the value based on the type of trap.
Specific Type	The specific trap type indicates the specific trap as defined in an enterprise-specific MIB. If the Generic type value is 6 then, this field shows a value greater than 0. If the generic type value is a value other than 6, then the field shows a value 0. This field can have values from 0 to 2147483647.
Message	This is a text field. By default, this field will always contain the Varbinds in the Trap PDU. This can be substituted with text.
Severity	This field shows the Severity or the intensity of the trap. They could be 0-All, 1-Critical, 2-Major, 3-Minor, 4-warning, 5-Clear and 6-info.
Entity	The source IP address from which the Trap was sent is depicted 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 depicts 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

The 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 is 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 emits 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, ISoIP, 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 19 SNTP client configuration

Chapter contents

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 231)

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 225)
- Defining SNTP client operating mode (see page 225)
- Defining SNTP local UDP port (see page 226)
- Enabling and disabling the SNTP client (see page 227)
- Defining the SNTP client anycast address (see page 228)
- Defining SNTP client constant offset to GMT (see page 227
- Enabling and disabling local clock offset compensation (see page 229)
- Defining SNTP client poll interval (see page 227)
- Showing SNTP client related information (see page 230)
- Debugging SNTP client operation (see page 230)

Selecting SNTP time servers

This procedure describes how to select a primary and secondary SNTP time server

Mode: Configure

Step	Command	Purpose
1	***node***(cfg)#**sntp-client server primary** *ip-address*	Enters the SNTP primary server IP address
2	***node***(cfg)#**sntp-client server secondary** *ip-address*	Enters the SNTP secondary server IP address

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	***node***(cfg)#**sntp-client operating-mode {unicast | anycast | multicast}**	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 the Section “Defining the SNTP Client Anycast Address” 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** *number*	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 as 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	***node***(cfg)#[no] sntp-client	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 in 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	***node***(cfg)#sntp-client poll-interval *value*	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	***node***(**cfg)#sntp-client gmt-offset** *offset*	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** *ip-address* **{port |** *port-number***}**	Set the anycast-address to *ip-address* a designated local broadcast or multicast group address to which a request is sent. In addition an explicit SNTP server *port-number* 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 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 16 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 severs. It is probably safest to use the IP addresses instead of the domain names.

Table 16. 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.bldrdoc.gov	132.163.4.101	1, 4	Colorado
time-b.timefreq.bldrdoc.gov	132.163.4.102	1	Colorado

Table 16. Time servers operated by NIST (Continued)

Server Name	IP Address	Note	Location
time-c.timefreq.bldrdoc.gov	132.163.4.103	1	Colorado
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 20 DHCP configuration

Chapter contents

Introduction

This chapter provides an overview of the Dynamic Host Configuration Control Protocol (DHCP) and describes the tasks involved in configuring them. This chapter includes the following sections:

- DHCP-client configuration tasks (see page 237)
- DHCP-server configuration tasks (see page 240)

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. This 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 provides other clients on the LAN side with IP addresses and other configuration information. DHCP-server and DHCP-client are illustrated in figure 49.

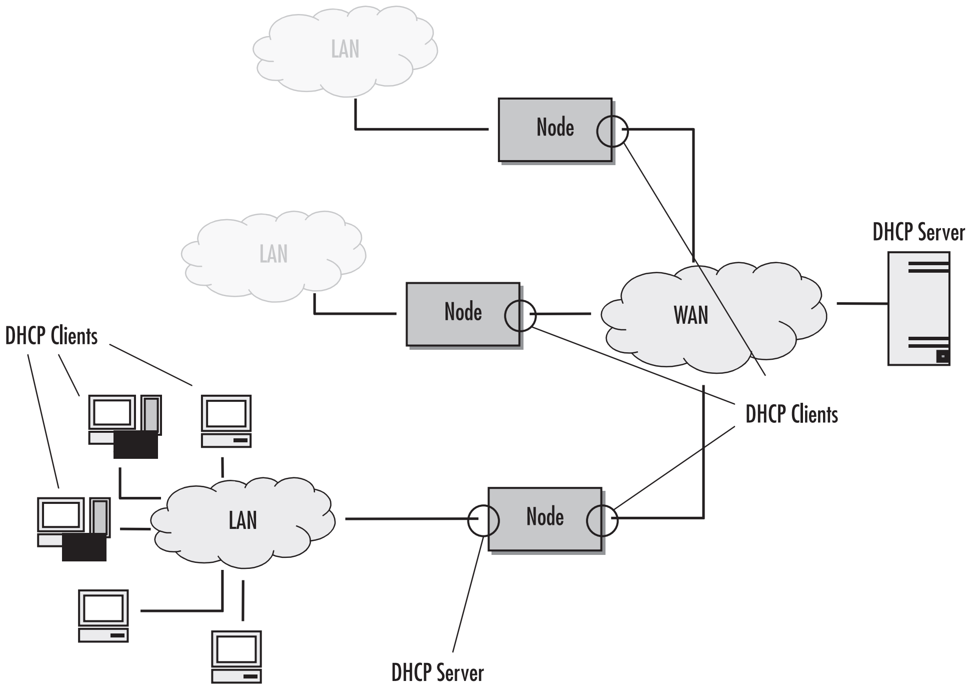


Figure 49. DHCP-client and DHCP-server on the SmartNode

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 238)
- Get debug output from DHCP-client (see page 239)
- 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	***node***(**ctx-ip)[router]#interface** *name*	Creates an IP interface with name *name* and enters 'configure' configuration mode
2	***node***(**if-ip)[***name***]#ipaddress dhcp**	Enables the DHCP-client on this IP interface. (See note)
3	***node***(**if-ip)[***name***]#show dhcp-client**	Displays status information about the DHCP-client. E.g. 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 has 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 has 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 242)
- Check DHCP-server configuration and status (see page 243)
- Get debug output from the DHCP-server (see page 243)

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** *name*	Enter DHCP-server profile mode
2	***node*(pf-dhcps)[name]#network** *ip-address ip-mask*	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** *ip-address-from ip-address-to*	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** *default-router-ip-address*	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** *time* **days|hours|minutes**	Defines the time a lease is valid DHCP Option 51
6 (optional)	***node*(pf-dhcps)[name]#[no] domain-name** *domain-name*	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** *domain-name-server-ip-address*	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** *netbios-name-server-ip-address*	Typical installation use *h-node* 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 b-node|h-node|m-node|p-node**	Defines the NetBIOS node type (b: ???, h: hybrid, m: ???, p: ???) DHCP Option 46
10 (optional)	***node*(pf-dhcps)[name]#[no] bootfile** *boot-file-name*	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** *next-server-ip-address*	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 *name*
2	***node*(ctx-ip)[router]#[no] dhcp-server**	Enables / disables DHCP-server
3	***node*(ctx-ip)[router]#dhcp-server clear-lease { all | *ip-address* }**	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 21 PPP configuration

Chapter contents

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 et al.. 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)
- PPP over ISDN (not available yet)

Figure 50 illustrates the elements involved in the configuration of PPP. The elements required to configure PPP over Ethernet are located in the upper left corner. The elements for PPP over serial link are in the lower left corner, for PPP over ISDN in the middle and the lower right corner.

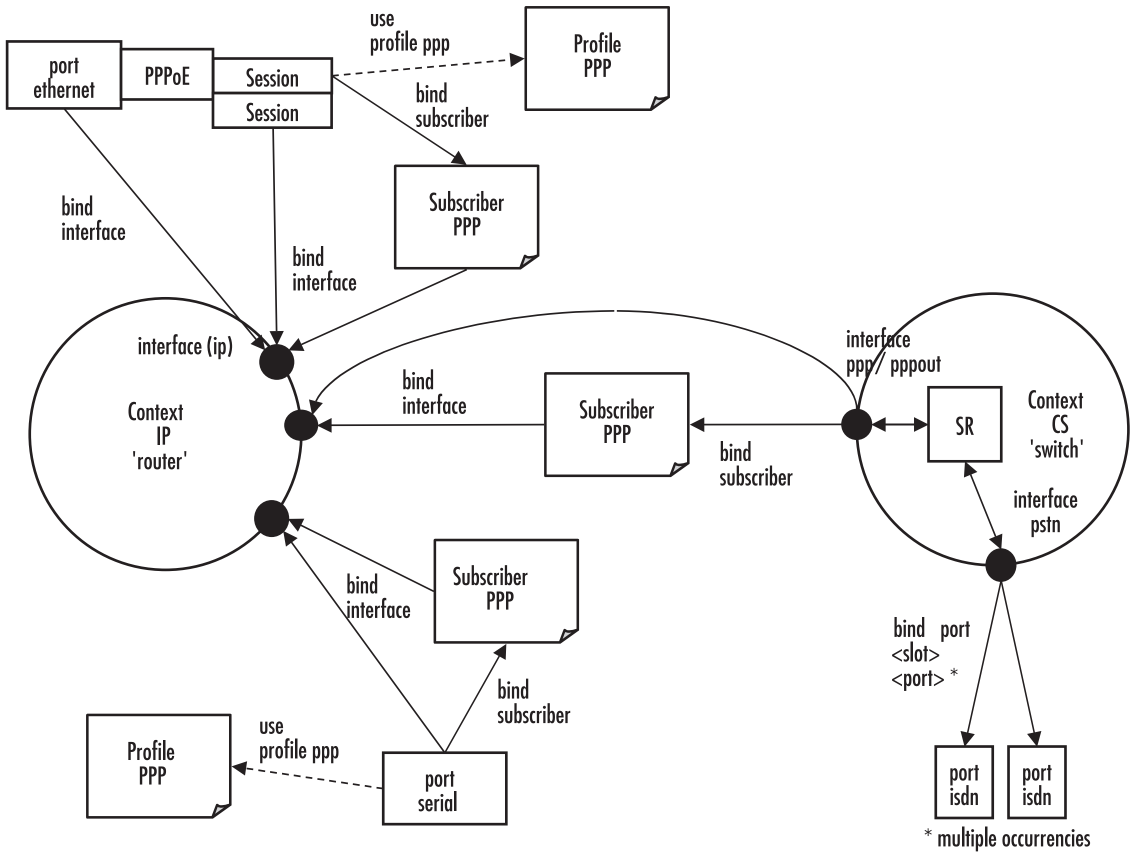


Figure 50. PPP configuration overview

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 249)
- Configuring a PPPoE session (see page 250)
- Configuring a serial port for PPP (see page 252)
- Creating a PPP interface within the CS context (not available yet)
- Creating a PSTN interface for PPP dial-in / dial-out (not available yet)
- Creating a PPP profile (see page 253)
- Displaying PPP configuration information (see page 254)
- Debugging PPP (see page 256)

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 11, "NAT/ NAPT configuration" on page 119 on page 119 for more information about NAPT).

This procedure describes how to create an IP interface for PPP

Mode: Context IP

Step	Command	Purpose
1	***node*(ctx-ip)[router]#interface *name***	Creates the new interface *name*, which represents an IP interface.
2	***node*(if-ip)[name]#point-to-point**	Only defines what route is entered into the IP routing table: • 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	***node*(if-ip)[name]#ipaddress unnumbered** or	The PPP remote peer offers an IP address for the IP interface. The IP interface adopts this IP address
	***node*(if-ip)[name]#ipaddress dhcp** or	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.
	***node*(if-ip)[name]# ipaddress** *ip-address netmask*	The IP interface requests from the PPP remote peer to use the IP address *ip-address*. 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 *netmask* specifies the size of the subnet in case 'no point-to-point' is configured
4 (optional)	***node*(if-ip)[*name*]# [no] tcp adjust-mss { rx I tx } { mtu I** *mss* **}**	Limits to the MSS (Maximum Segment Size) in TCP SYN packets to *mss* 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. 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. IP packets must therefore 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. It is recommended to use '**mtu**' inbound and outbound.

Step	Command	Purpose
5 (optional)	***node*(if-ip)[name]#use profile napt** *name*	Assigns the NAPT profile *name* to applied to this IP interface. See 11, "NAT/NAPT configuration" on page 119 on page 119 to learn how to create a NAPT profile.

Example: Create an IP interface for PPP

The procedure below creates an IP interface which 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	***node*(cfg) # subscriber ppp name**	Creates the new subscriber *name*, which contains the authentication settings.
2	***node*(subscr)[name]# dial {in|out}**	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: • PPP over Ethernet: 'dial out' • PPP over Serial: 'dial in'
3	***node*(subscr)[name]# [no] authentication { (chap pap) | {chap|pap} }**	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*]	Sets the credentials to be provided during the authentication procedure: the user name *user* and the password *password*. The keywords 'inbound' and 'outbound' define the direction of authentication: • '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. The following restrictions apply to the direction of authentication: • - 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 247.

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 which allows to run 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** *slot port*	Enters Ethernet port configuration mode for the interface on *slot* and *port*
2	***node*** **(prt-eth)[slot/port]# encapsulation {ip|pppoe|multi}**	Defines the payload type(s) to be used on the Ethernet: • '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** *name* **[router]**	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** *name*	Creates PPPoE Session with the name *name*
7	***node***(**pppoe)[slot/port]# [no] bind interface** *name* **[router]** or ***node*** **(pppoe)[slot/port]# [no] bind subscriber** *name*	Binds the PPPoE Session directly to the IP interface *name* in case no authentication is required Binds the PPPoE Session to the PPP subscriber *name* in case authentication is required
8 (optional)	***node*** **(pppoe)[slot/port]# [no] use profile ppp** *name*	Assigns a PPP profile other than the default profile to this PPPoE Session
9 (optional)	***node***(**session)[name]#service** *Service-Name*	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** *AC-Name*	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** *slot port*	Enters the configuration mode for the serial port on *slot* and *port*
2	***node***(**prt-ser)[slot/port]# [no] encapsulation { framerelay|ppp }**	Sets the encapsulation to 'ppp'
3	***node*** **(prt-ser)[slot/port]# [no] bind interface** *name* **[router]** or ***node*** **(prt-ser)[slot/port]# [no] bind subscriber** *name* or ***node*** **(prt-ser)[slot/port]# [no] bind subscriber authentication { chap pap | { chap | pap } }**	Binds the serial port directly to the IP interface *name* in case no authentication is required Binds the serial port to the PPP subscriber *name* 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 *name* and enters the PPP profile configuration. The profile 'default' already exists.
2 (optional)	***node*(pf-ppp)[name]#mtu min** *min* **max** *max*	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** *min* **max** *max*	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.

Step	Command	Purpose
4 (optional)	***node*(pf-ppp)[name]#[no] van-jacobson {compression|decompression} max-slots** *max-slots*	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. *max-slots* 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**	Gives the best overview of all PPP related configuration information. The following parts are of interest: • profile ppp default • profile ppp *name* • interface *name* • subscriber ppp *name* • port ethernet *slot port* • session *name*
2	***node*(cfg) #show subscriber ppp [** *name* **]**	Displays configuration information of the PPP subscriber *name* or of all PPP subscribers
3	***node*(pf-ppp)[name]#show profile ppp {** *name* **| default }**	Displays the PPP profile *name* 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): inalp/inalp
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* **]**	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'. level specifies to level of details displayed (1..4, default is 1).
2	***node***(cfg) **#show ppp networks [** *level* **]**	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'. 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. *level* specifies to level of details displayed (1..4, default is 1).
3	***node***(cfg) **#show pppoe [** *name* **]**	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'. *level* specifies to level of details displayed (1..4, default is 1).
4	***node***(cfg) **#show port interface** *name*	Displays status and configuration information of the IP interface at which a PPP connection terminates. Check whether state of the interface is 'OPENED'. 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'.
5	***node***(cfg) **#show port ethernet** *slot port* or ***node***(cfg) **#show port serial** *slot port*	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).
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:                  inalp
    Password:            inalp
    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 22 VPN configuration

Chapter contents

Introduction

This chapter describes how to configure the VPN connections between two SmartNodes or between a SmartNode and a third-party device.

This chapter includes the following sections:

- VPN Configuration Task List
- Examples

A Virtual Private Network (VPN) is a private data network that makes use of 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 explain 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 Bit over the packet to be protected (see RFC 2403).
- HMAC-SHA1-96:is a combination of the Keyed-Hashing for Message Authentication (HMAC) and the Secure Hash Algorithm version 1 (SHA1). It requires an authenticator of 160 Bit length and calculates a hash of 96 Bit 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 does not offer adequate security any longer because of its short key length. A key length of at least 100 Bit is recommended. The AES algorithm is very efficient and allows the fastest encryption. AES with a key length of 128 Bit 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. The IPsec peers must therefore be the endpoints of the communication.

  A secure connection of 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 the endpoints of the communication.

  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 into the startup-configuration of the SmartNode and its peer, as a hexa-decimal string.

Note The key life-time of a DES-key ranges between 3 hours and 3 days depending on the processing hardware applied to 'reverse engineering' the key. DES-Keys must be manually updated in constant intervals in order to maintain security. The life-time of an AES- or 3DES-key is close to the infinity compared to a human life.

The Internet Key Exchange (IKE) protocol is not supported yet.

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** *name*	Creates the IPsec Transformation Profile *name*
2 optional	***node*(pf-ipstr)[*name*]#esp-encryption { aes-cbc | des-cbc | 3des-cbc } [*key-length*]**	Enables encryption and defines the encryption algorithm and the key length Supported key lengths see section "Encryption" on page 264
3 optional	***node*(pf-ipstr)[*name*]#{ 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 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** *name*	Creates the IPsec Policy Profile name
2	***node***(**pf-ipstr)[name]#use profile ipsec-transform** *name*	Selects the IPsec Transformation Profile to be applied
3 optional	***node***(**pf-ipstr)[***name***]#session-key** **{ inbound | outbound }** **{ ah-authentication | esp-authentication | esp-encryption }** *key*	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. 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 264 and "Encryption" on page 264) or explicit specification. 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.
4	***node***(**pf-ipstr)[***name***]#spi** **{ inbound | outbound } { ah | esp }** *spi*	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 $1..2^{32}-1$. SPIs must be available for encryption and/or authentication as specified in the respective IPsec Transformation Profile. 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.
5	***node***(**pf-ipstr)[***name***]#peer** *ip-address*	Sets the IP address of the peer **Note** The peers of the secured communication must have static IP address. DNS resolution is not available yet.
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 ACL (Access Control List) Profile in 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. The respective ACLs must therefore permit the encrypted/authenticated and the plain traffic.

For detailed information on how to set-up ACL rules see chapter 17, "Access control list configuration" on page 193.

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 *name* 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	***node*(cfg)#context ip router**	Enter IP context
2	***node*(ctx-ip)[**router**]#interface** *if-name*	Create/enter the IP interface *if-name*
3	***node*(if-ip)[***if-name***]# use profile acl** *name* **out**	Activate the outgoing ACL profile *name*
4	***node*(if-ip)[***if-name***]#context ip router**	Enter IP context
5 optional	***node*(ctx-ip)[**router**]#route** *remote-network-address remote-network-mask if-name* **0**	Creates a route for the remote network that points the above IP interface *if-name* This setting can be omitted if the default route already points to this IP interface or to a next hub reachable via this IP interface and if 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	***node*(cfg)#show profile ipsec-transform**	Displays all IPsec Transformation Profiles
2 optional	**node(cfg)#show profile ipsec-policy-manual**	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	***node*(cfg)#debug ipsec**	Enables IPsec debug monitor
2 optional	***node*(cfg)#show ipsec security-associations**	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 Pro-
file 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
!
...
```

Rest 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
...
```

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 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
!
...
```

Rest of the configuration, see above, just change the name of the IPsec Policy Profile in the ACL profile 'VPN_Out'

Chapter 23 **CS context overview**

Chapter contents

Introduction

This chapter gives an overview of the SmartWare circuit-switching (CS) context and its associated components and describes the tasks involved in its configuration. If you want understand the CS entity configuration you should read this chapter. You will get a basic understanding of how to set up a SmartNode to support voice calls.

The sections describe the configuration steps needed to perform a complete voice connectivity configuration, and refer specifically to the other chapters in which the respective configuration topic is explained in more detail. To understand the given information in the following sections it is necessary that you have read and understood the SmartWare configuration concepts as described in chapter 2, "Configuration concepts" on page 29.

The CS context in SmartWare is a high level conceptual entity that is responsible for all aspects of circuit signaling, switching and emulation. The CS entity comprises the Context CS itself, CS Interfaces, ISDN Ports, Tone-Set Profiles, ISoIP and H.323 Gateways and VoIP Profiles as circled by the area enclosed by a dashed line in figure 51:

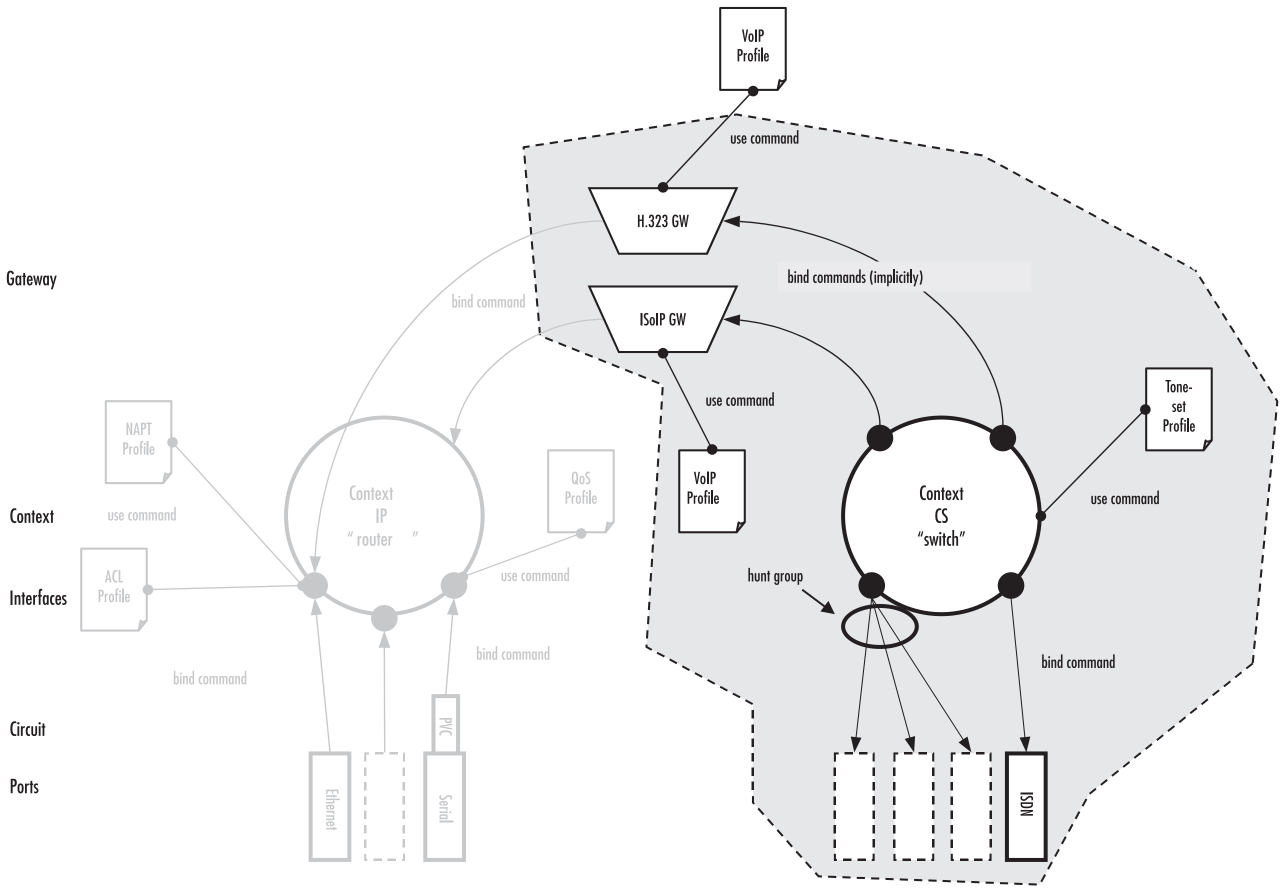


Figure 51. CS context configuration components

The job of the CS context and its associated components is to route and establish a voice call. For example the signaling for dial-up circuits is routed and the corresponding voice call circuits are switched between PSTN interfaces and via VoIP interfaces to the VoIP gateways and the IP Context. This is explained in more detail in the configuration task "Configure call routing" on page 288.

CS context configuration task list

The information that is needed for the CS entity configuration is distributed among several configuration tasks, depending on its logical content.

For example, all related information that pertains to call routing is described in configuration task "Configure call routing" on page 288. These configuration steps themselves can be described in one or more other chapters. Thus, to configure call routing you have to refer to chapter 24, "CS interface configuration" on page 295 and chapter 25, "Session router configuration" on page 311.

This chapter shows you the correlation between the several CS configuration components. We recommend that you perform the CS context configuration in the order that is described below. Many of the parameters have default values which do not need to be changed. This means, that you generally do not need perform all of the described configuration tasks. In such cases it is stated in the text that you can skip the respective configuration task.

- Plan the CS configuration
- Configure general CS settings (see page 288)
- Configure call routing (see page 288)
- Configure dial tones (advanced) (see page 282)
- Configure Voice over IP settings (advanced) (see page 283)
- Configure ISDN ports (see page 283)
- Configure an ISoIP VoIP connection (see page 283)
- Configure an H.323 VoIP connection (see page 284)
- Activate CS context configuration (see page 284)

Plan the CS configuration

There are many possible policies and factors that may 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 (ISoIP or H.323); gatekeeper settings in the case of H.323
- Number and type of interface cards installed in your SmartNode(s)
- Call routing

For an example see figure 52 on page 279, which depicts a remote office in an enterprise network. The example concentrates 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 ISoIP with a codec G.711. A call is routed to the IP backbone as well as to the Public ISDN network depending on its prefix and number length. This implies the following CS configuration:

Because we are connected to the Public Switched Telephone Network, we get the clock-source from the corresponding ISDN port and simultaneously synchronize the system time of the SmartNode to the ISDN time. (Described in section "Configure general CS settings" on page 280). In general, please be careful from which location you get your clock source. If your clock for packaging the ISDN voice frames is not synchronized with the remote ISDN clock, this may result in bit errors. In case of fax applications not one page could be transmitted.

- We need two ISDN ports, the first one for the ISDN phone, the second one for the public ISDN network. This is described in section 'Configure ISDN Ports'.
- Furthermore we need two PSTN interfaces, each bound to an ISDN port. (This is described in section "Configure call routing" on page 288)
- To use ISoIP we need an ISoIP interface. (This is described in section "Configure call routing" on page 288)
- To support call routing we have to configure Session Router routing tables and the ISoIP and PSTN interfaces. (This is described in section "Configure call routing" on page 288). Calls are routed:
  - from the ISDN phone with number 1xx-5xx, or with prefix 0041 for CH, to main office with a fallback to PSTN
  - All others are routed from the ISDN phone to PSTN
  - from PSTN or main office to the ISDN phone
- We have to specify codec G.711 in the ISoIP gateway. This is described in section "Configure an ISoIP VoIP connection" on page 283
- Furthermore we need in the IP context two Ethernet ports and their corresponding IP interfaces.

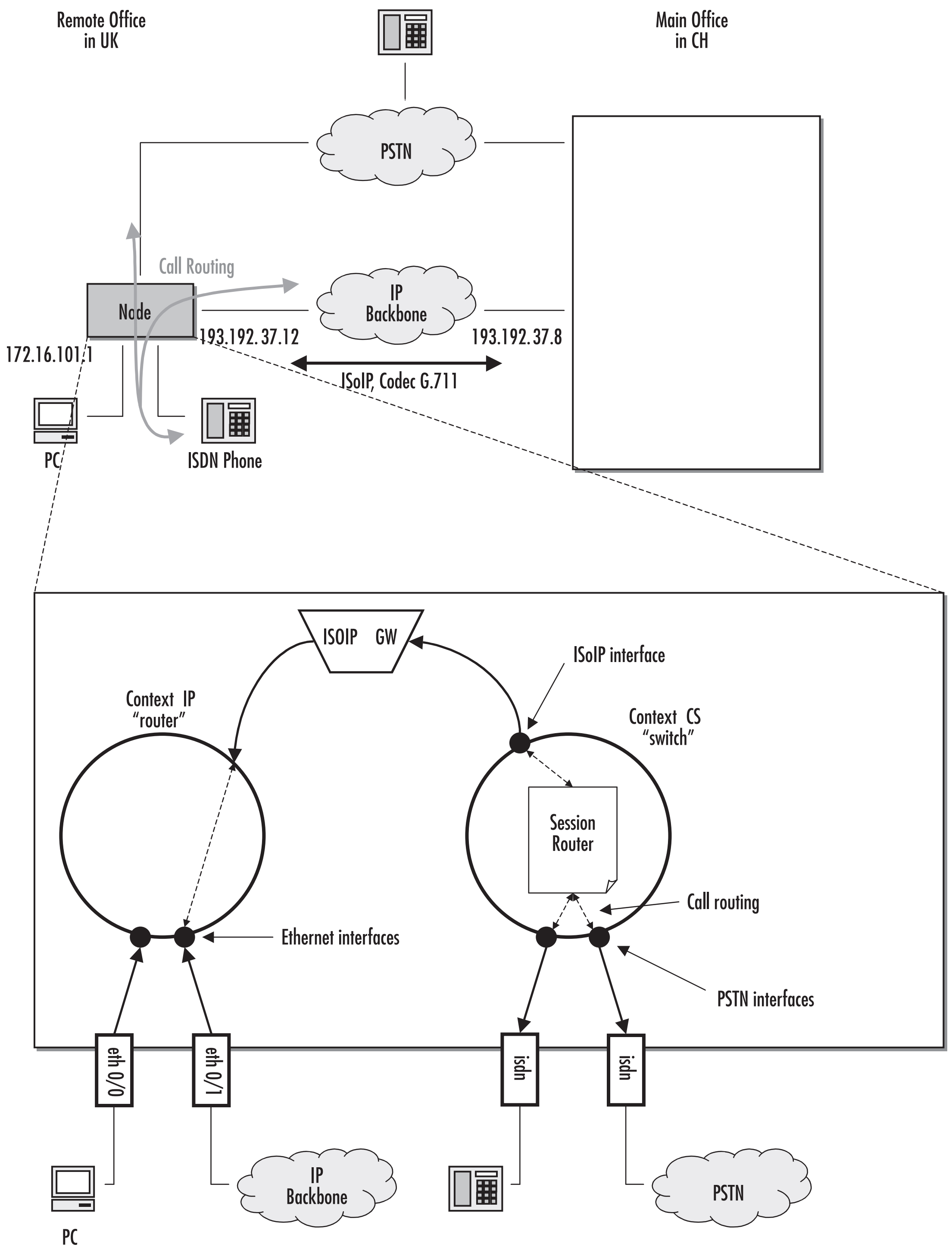


Figure 52. Remote office in an enterprise network

You must not start configure the CS context and its components until you have finished planning your voice environment. The following sectons explain how to realize 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 9, "IP context overview" on page 101.

Configure general CS settings

There are several parameters which can not be collected into one specific configuration task, because they are independent of the rest of the CS context configuration and apply mostly to a whole interface card or even to the whole SmartNode.

In most cases the default value is suitable, so that you do not need to perform this configuration tasks.

- **Configure clock source**—The packaging of the ISDN voice frames needs a reference clock. It is possible to generate this reference clock internally or get it from an external device (e.g. public ISDN). If you have a connection to a public ISDN this is the normal case.
- **Select PCM law**—The PCM law-select specifies the voice characteristic compression curve. Two values are possible: "a-Law" and "µ-Law". "a-Law" is used in Europe, and "µ-Law" in USA.
- **Synchronize to ISDN time**—The internal time of the SmartNode can be synchronized to the time on the public ISDN.
- **Set bypass mode**—The SmartNode contains a hardware bypass, which connects two ISDN ports in case of power failure. You can manually enable it on a running system.

This procedure describes how to set the general CS parameters

Mode: System

Step	Command	Purpose
1	***node*(sys)#no bypass-mode**	Turn off the hardware bypass.
2	***node*(sys)#clock-source internal** or ***node* (sys)#clock-source** *slot-number port-number*	Generate the reference clock internally or specify a specific port to receive the reference clock.
3	***node*(sys)#synchronize-to-isdn-time**	Enable SmartWare to get the actual system time from public ISDN.
4	***node*(sys)#ic voice** *slot-number*	Change to mode ic_voice.
6	***node*(ic-voice)[*slot-number*]#pcm law-select { aLaw | uLaw }**	Select the PCM aLaw for Europe or uLaw for USA

Example: The following example configures the general CS parameters

```
SN>enable
SN#configure
SN(cfg)#system
SN(sys)#no bypass-mode
SN(sys)#clock-source 1 0
SN(sys)#synchronize-to-isdn-time
SN(sys)#ic voice 1
SN(ic-voice)[1]#pcm law-select aLaw
SN(ic-voice)[1]#exit
```

Configure call routing

Calls through a SmartNode can be routed according to a set of routing criteria. The entity that manages call routing is called Session Router. Calls are routed from one CS interface to another. The Session Router determines the destination interface for every incoming call. It supports complex call routing and number manipulation functions. See chapter 25, "Session router configuration" on page 311.

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 three types of CS interfaces: PSTN, H.323 and ISoIP interface. The differences between the several interfaces are explained later.

As an example, figure 53 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 left-hand phone over the PSTN interface directly to an ISoIP interface. Once it has passed the IP context and the IP Network, the other SmartNode—from the ISoIP interface to the PSTN interface and then over the ISDN port to the B-party phone—routes the call.

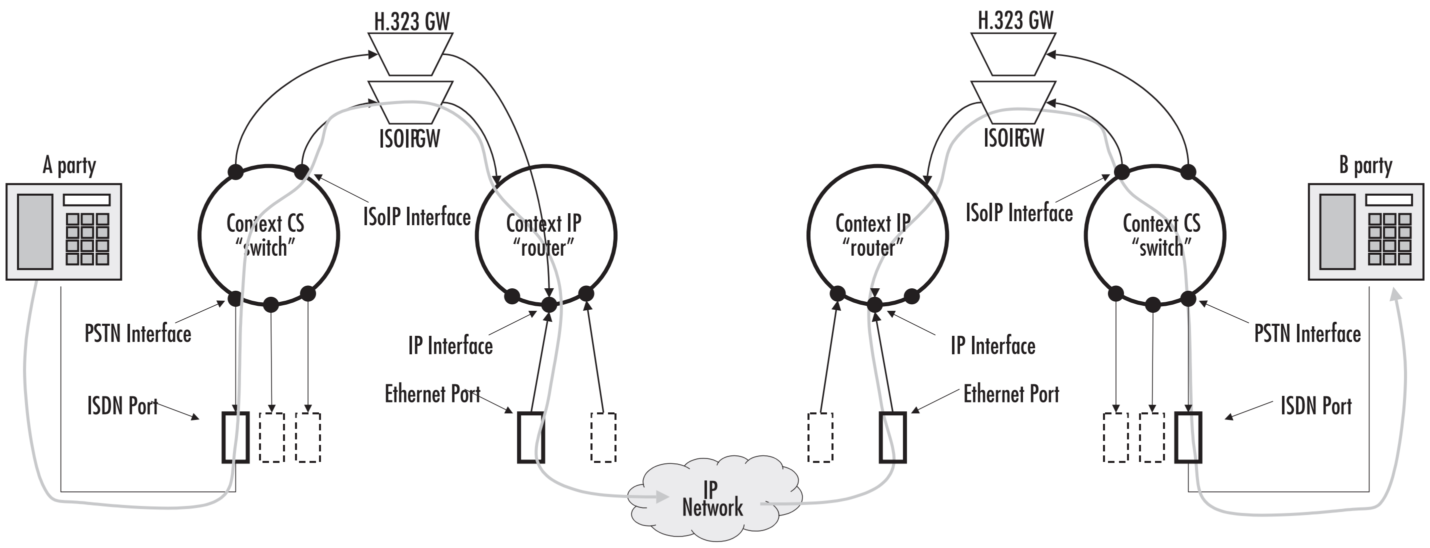


Figure 53. Direct call routing from one SmartNode to another

Because call routing occurs only in the **CS context**, in future figures the IP context is omitted. For configuring call routing you have to create the **CS interfaces** and the session router tables as described in the chapters below. For simple call routing directly from one interface to another you can even omit creating session router tables.

Create and configure CS interfaces

SmartWare currently supports one instance of the CS context. The name of this 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 session 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. For examples and information on how to create CS interfaces please refer to chapter 24, “CS interface configuration” on page 295.

Specify call routing

As described above, for basic call routing you can omit creating session router tables. SmartWare offers two levels of call routing:

- Basic interface routing
- Advanced session routing

Basic interface routing allows you to forward all incoming calls on a CS interface directly to a destination CS interface. An optional fallback interface can also be specified. The Session Router allows you to route calls to all available CS interfaces, based on a number of criteria such as calling number, destination number and ISDN bearer capability.

We recommend that you first carefully consider what interfaces and session router tables are required to achieve your goals on a sheet of paper and then start with creating and configuring CS interfaces and set up the session router tables.

To configure Basic Interface Routing refer to chapter 24, “CS interface configuration” on page 295. Other topics such as *digit collection* or *specifying port addresses*, which belong to call routing are explained in this chapter.

To configure *Advanced Session Routing* in relation to the session router tables refer to chapter 25, “Session router configuration” on page 311. In this chapter the difference between *Basic interface routing* and *Advanced session routing* is described in more detail. If you are not familiar with call routing it is suggested that you read the respective sections in this chapter first, before you plan the CS context configuration.

Configure dial tones

SmartWare supports country-specific configurable in-band dial tones that are played for specific events, for example alerting, dialing or busy. The tones are configured in tone-set profiles, and these profiles are used either from the CS context or from a specific PSTN interface.

Normally tone-sets are used by the CS context, this means that all PSTN interfaces use the tone-set profile which is used by the CS context. In case a specific PSTN interface needs to use other dial tones, a different tone-set profile can be used directly from the PSTN interface. To configure the dial tones and apply then to the CS context or interface see chapter 25, “Session router configuration” on page 311.

If no tone-set 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.

Configure Voice over IP parameters

In SmartWare there are many parameters that is possible to configure which can affect a voice over IP connection. SmartWare supports two different protocols that transmit voice packets over IP, the Patton Electronics Co. developed protocol ISoIP and the generally defined H.323 protocol.

The Voice over IP (VoIP) parameters are configured in the VoIP profile. A VoIP profile is used by a ISoIP or H.323 gateway. All calls going through that gateway use the settings in the VoIP profile. It is possible, for a specific ISoIP or H.323 interface, to overwrite some of the settings made in the VoIP profile, so that a call through this specific interface uses other settings. Parameters which are configured in the VoIP profile are:

- Filters
- DTMF relay
- Echo canceller
- Silence compression
- Voice volume
- Dejitter buffer

Parameters which can be overwritten on a CS interface are:

- DTMF relay
- Echo canceller
- Silence compression
- Voice volume
- Dejitter buffer

For configuring the general VoIP parameters refer to chapter 30, "VoIP profile configuration" on page 377. Some settings can greatly affect the voice quality perceived by the user and the bandwidth requirements of VoIP connections. Therefore be sure you understand the meaning of the commands prior to changing any settings. Most of the default values of this parameters are suitable, so that you generally do not need to perform this configuration tasks.

Configure ISDN ports

ISDN ports represents physical ports on the SmartNode. The configuration of the ISDN ports depends on the port type (BRI or PRI), and on the connected voice device. To configure the ISDN ports refer to chapter 30, "VoIP profile configuration" on page 377.

Configure an ISoIP VoIP connection

To configure an ISoIP connection you need to specify the voice codec selection and the call signaling method.

The codec determines the voice compression technique. The codec which is used by default for a ISoIP voice connection is specified in the ISoIP gateway. All calls through this ISoIP gateway use these codec. If there is a specific ISoIP interface which needs to use another codec than the default codec specified in the ISoIP gateway, it is possible to overwrite the default codec for this interface. For information how to configure the default codec selection for an ISoIP voice connection refer to chapter 20, "DHCP configuration" on page 235.

The call signaling method specifies the call setups to the destination SmartNode. In ISoIP there is one configuration methods, 'direct call signaling'. This means that every ISoIP call needs the IP address of the destination SmartNode. Therefore in every ISoIP interface a remote IP address is specified. For examples and information on how to configure direct call signaling for ISoIP voice connections, refer to chapter 20, "DHCP configuration" on page 235.

Configure a H.323 VoIP Connection

To configure a H.323 connection you have to specify the voice codec selection, the call signaling method and Q.931 tunneling.

Configuring the voice codec for an H.323 connection differs from the procedure used for ISoIP. In the H.323 gateway a list of all possible codecs which we want to use we have to specified, and in the H.323 interface one of these listed codecs we have to specified. If no codec is specified in the H.323 interface, during a call setup the first codec listed in the H.323 gateway which matches with the remote SmartNode is taken. For information how to configure the codecs for a H.323 connection refer to chapter 20, "DHCP configuration" on page 235.

In H.323 there are two call signaling methods, 'direct call signaling' and 'gatekeeper routed call signaling'. Direct call signaling works in the same way as ISoIP. 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 20, "DHCP configuration" on page 235. To configure gatekeeper routed call signaling on H.323 voice connections refer to chapter 20, "DHCP configuration" on page 235.

Q.931 tunneling is able to support ISDN services and Q.SIG over an IP network. In ISoIP Q.931 tunneling is always enabled. To configure Q.931 tunneling on a H.323 connection, refer to chapter 20, "DHCP configuration" on page 235.

Activate CS Context Configuration

After configuring the CS context and all its components the configuration must be activated. This includes binding the PSTN interfaces to the ISDN ports and enabling the gateways, ISDN ports and the CS context.

In order to become functional, each PSTN interface must be bound to one or more ISDN ports from which it receives incoming calls and to which it forwards outgoing calls. Note that there is a difference between IP interfaces and Ethernet ports. In the IP context the **Ethernet port is bound** to the IP interface, in the CS context the **PSTN interface is bound** to the ISDN port! After binding to become active the ISDN port must be enabled.

To bind the PSTN interfaces to ISDN ports refer to chapter 20, "DHCP configuration" on page 235 and to enable the ISDN ports refer to chapter 27, "ISDN port configuration" on page 341. Likewise the ISoIP or H.323 gateway must be enabled. Additionally the H.323 gateway must be bound to a specific IP interface. For more information refer to chapter 27, "ISDN port configuration" on page 341.

In order to become active the CS context must be enabled. When recovering from the shutdown status, the CS context and Session Router configuration is checked and possible errors are indicated. The Session Router debug monitor can be enabled to show the loading of the CS context and Session Router configuration. SmartWare offers a number of possibilities to monitor and debug the CS context and session router configurations. For example the Session Router debug monitor enables you follow the sequence of tables and functions examined by the Session Router for each call setup. Refer to chapter 27, "ISDN port configuration" on page 341 for an introduction to the configuration debugging possibilities in SmartWare.

Note You can modify the configuration at runtime, but changes will not be active immediately. It is not necessary to shutdown the CS context prior to making any configuration changes, but if the CS context is not activated with the command **no shutdown** the newly created or changed configuration is not loaded!

There are several possibilities to show the actual CS context configuration. Note that some of the show commands display only the actual configuration. It could be that you have changed your configuration and it is not displayed because the configuration is reloaded. For more information on the show command refer to the respective configuration chapters or to chapter 31, "VoIP debugging" on page 395.

This procedures describes how to show the CS context configuration, enable the Session Router debug monitor and activate the CS context

Mode: Context CS

Step	Command	Purpose
1	***node***(ctx-cs)[switch]#**show context cs config** *level*	Show the CS context configuration. *Level* could be 0..5. Level 0 shows less, level 5 shows all information.
2	***node***(ctx-cs)[switch]#**debug session-router**	Enable the session-router debug monitor
3	***node***(ctx-cs)[switch]#**no shutdown**	Enable the CS context, checks the interface and session router configuration

Example: Enable CS context

The following example shows how to disable the current CS context configuration, how to enable the Session Router debug monitor and how to enable the CS context. It also shows the output from the Session Router debug monitor.

Note The interface *callapp* is a default interface, that is always defined and cannot be deleted.

```
SN(cfg)#show context cs config 5
Following session-router configuration sets are available:
  switch
    Interfaces:
      H323_IF
      ISDN
SN(cfg)#debug session-router 5
SN(cfg)#context cs
SN(ctx-cs)[switch]#no shutdown
SN(ctx-cs)[switch]#00:01:39  SR    > Loading interfaces...
00:01:39  SR    > Resolving interface references interfaces...
00:01:39  SR    > Classifier is resolving interface references...
00:01:39  SR    > Loading session router tables...
00:01:39  SR    > Resolving routing table references within routing tables...
00:01:39  SR    > Resolving interface references within routing tables...
00:01:39  SR    > Resolving routing table references within interfaces...
00:01:39  SR    > Classifier is resolving session router         references...
```

```
00:01:39  SR    > Loading and linking complete...
00:01:39  SR    > Following voice interfaces have been loaded:
00:01:39  SR    >   callapp
00:01:39  SR    >   H323_IF
00:01:39  SR    >   ISDN
00:01:39  SR    > Following routing tables have been loaded:
00:01:39  SR    > Following functions have been loaded:
00:01:39  SR    > Following number replacement tables have been loaded:

SN(ctx-cs)[switch]#
```

Example: Configure SmartNode in an enterprise network

Situation

Figure 54 on page 287 shows an enterprise network with a SmartNode 2300 series 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, therefore 'direct call signaling' is used. The voice codec used is G.723, therefore DTMF relay is enabled. Because no special dial tones have to be specified, the default tone-set profile is used.

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)

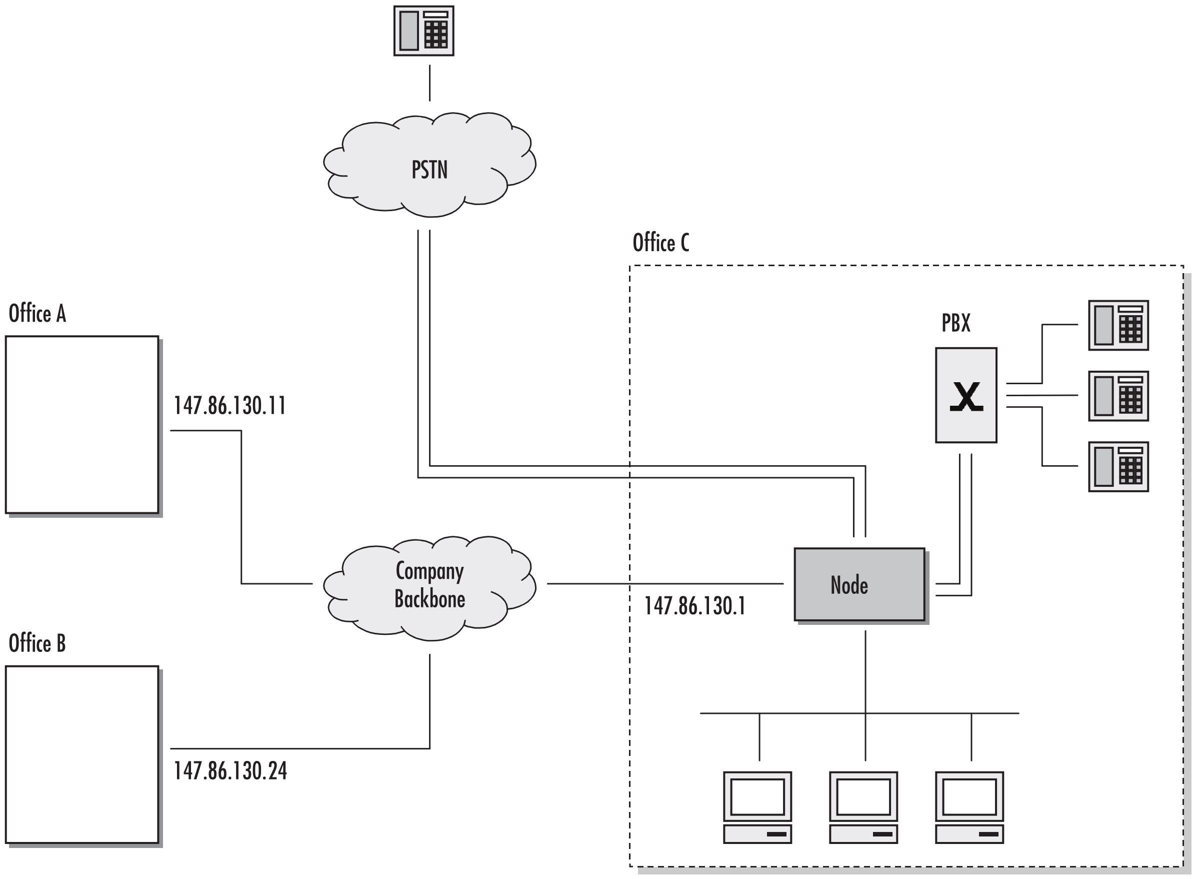


Figure 54. SmartNode in an Enterprise Network

Plan the CS context

The configuration described above implies the following configuration:

- It is very important to specify from where to get the clock source for the packaging of the ISDN voice frames. In our example we are connected to the PSTN network and get the clock source from the ISDN over the ISDN port 2/3.
- We synchronize the time of the SmartNode to the ISDN time. (Refer to section "Configure general CS settings" on page 280). To get useful system event logs it is very recommended to set the system time!
- We need four ISDN ports, two for the PSTN and another two for the PBX. (Refer to section "Configure ISDN ports" on page 283)
- Furthermore we need two PSTN interfaces. On each we bind two ISDN ports, which means, that we create a line hunt group on every PSTN interface. (Refer to section "Configure call routing" on page 288)
- 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 "Configure call routing" on page 288)
- We need a Session Router routing table to route the calls depending on the called party number. (Refer to section "Configure call routing" on page 288)

- We enable DTMF relay. (Refer to section "Configure Voice over IP settings" on page 289)
- We specify codec G.723, likewise we define H.323 'direct call signaling'. (Refer to section "Configure H.323 VoIP connection" on page 290)

Figure 55 illustrates the above-mentioned CS configuration

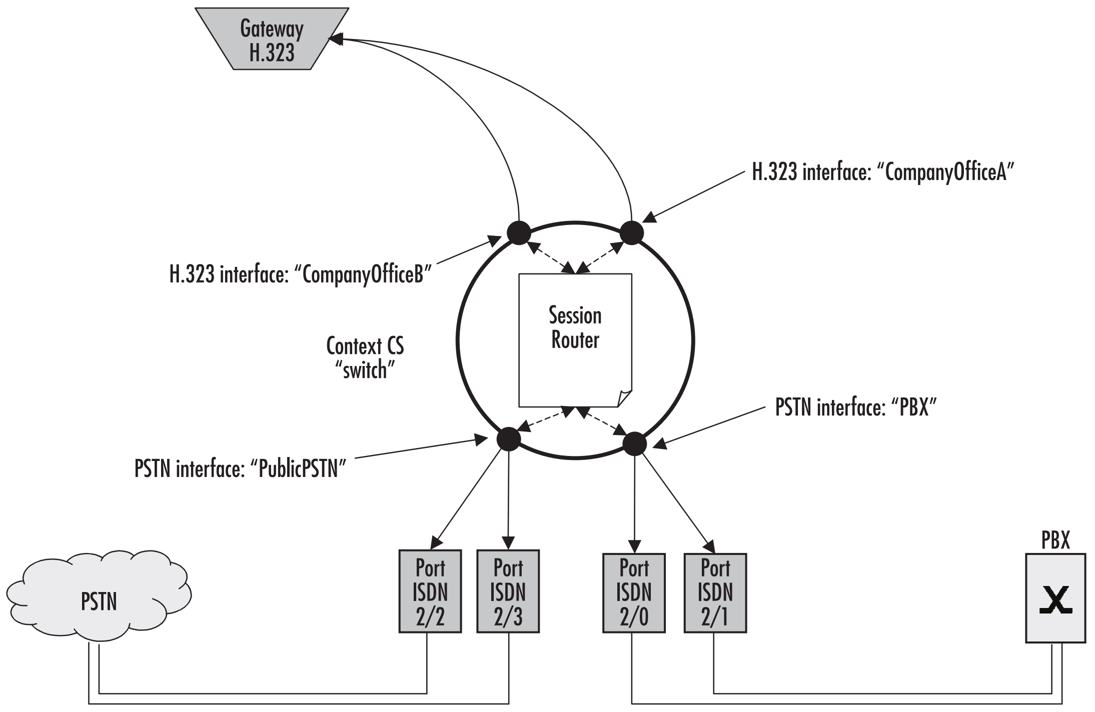


Figure 55. Configuration

Configure general CS settings

First we set clock-source to ISDN port 2/3 and enable synchronize to ISDN time.

```
SN>enable
SN#configure
SN(cfg)#system
SN(sys)#clock-source 2 3
SN(sys)#synchronize-to-isdn-time
SN(sys)#exit
SN(cfg)#
```

Configure call routing

Next we create the PSTN interfaces and configure call routing:

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface pstn PBX
SN(if-pstn)[PBX]#routing dest-table calledNumberRouting
SN(if-pstn)[PBX]#fallback dest-interface PublicPSTN
SN(if-pstn)[PBX]#exit
SN(ctx-cs)[switch]#interface pstn PublicPSTN
SN(if-pstn)[PublicP~]#routing dest-table calledNumberRouting
SN(if-pstn)[PublicP~]#exit
SN(ctx-cs)[switch]#
```

In addition we create the two H.323 interfaces and configure call routing:

```
SN(ctx-cs)[switch]#interface h323 CompanyOfficeA
SN(if-h323)[Company~]#routing dest-table calledNumberRouting
SN(if-h323)[Company~]#remoteip 146.86.130.11
SN(if-h323)[Company~]#exit
SN(ctx-cs)[switch]#interface h323 CompanyOfficeB
SN(if-h323)[Company~]#routing dest-table calledNumberRouting
SN(if-h323)[Company~]#remoteip 146.86.130.24
SN(if-h323)[Company~]#exit
SN(ctx-cs)[switch]#
```

Last we configure the Session Router:

```
SN(ctx-cs)[switch]#called-party CalledNumberRouting 1 dest-interface CompanyOfficeA
SN(ctx-cs)[switch]#called-party CalledNumberRouting 2 dest-interface CompanyOfficeB
SN(ctx-cs)[switch]#called-party CalledNumberRouting 5 dest-interface PBX
SN(ctx-cs)[switch]#called-party CalledNumberRouting default dest-interface PublicP-
STN
SN(ctx-cs)[switch]#exit
SN(cfg)#
```

Configure Voice over IP settings

Because we need G.723 as codec we enable DTMF relay:

```
SN(cfg)#profile voip H323VoIPProfile
SN(pf-voip)[H323VoI~]#dtmf-relay
SN(pf-voip)[H323VoI~]#exit
SN(cfg)#
```

Configure ISDN ports

Next is to configure the ISDN ports:

```
SN(cfg)#port isdn 2 0
SN(prt-isdn)[2/0]#down
SN(prt-isdn)[2/0]#channel-range 0 1
SN(prt-isdn)[2/0]#l2proto pp
SN(prt-isdn)[2/0]#l3proto pss1
SN(prt-isdn)[2/0]#max-channels 2
SN(prt-isdn)[2/0]#uni-side net
SN(prt-isdn)[2/0]#up
SN(prt-isdn)[2/0]#exit
SN(cfg)#
SN(cfg)#
SN(cfg)#port isdn 2 1
SN(prt-isdn)[2/0]#down
SN(prt-isdn)[2/0]#channel-range 0 1
SN(prt-isdn)[2/0]#l2proto pp
SN(prt-isdn)[2/0]#l3proto pss1
SN(prt-isdn)[2/0]#max-channels 2
SN(prt-isdn)[2/0]#uni-side net
SN(prt-isdn)[2/0]#up
SN(prt-isdn)[2/0]#exit
SN(cfg)#
SN(cfg)#
SN(cfg)#port isdn 2 2
SN(prt-isdn)[2/0]#down
SN(prt-isdn)[2/0]#channel-range 0 1
SN(prt-isdn)[2/0]#l2proto pp
SN(prt-isdn)[2/0]#l3proto dss1
SN(prt-isdn)[2/0]#max-channels 2
SN(prt-isdn)[2/0]#uni-side usr
SN(prt-isdn)[2/0]#up
SN(prt-isdn)[2/0]#exit
SN(cfg)#
SN(cfg)#
SN(cfg)#port isdn 2 3
SN(prt-isdn)[2/0]#down
SN(prt-isdn)[2/0]#channel-range 0 1
SN(prt-isdn)[2/0]#l2proto pp
SN(prt-isdn)[2/0]#l3proto dss1
SN(prt-isdn)[2/0]#max-channels 2
SN(prt-isdn)[2/0]#uni-side usr
SN(prt-isdn)[2/0]#up
SN(prt-isdn)[2/0]#exit
SN(cfg)#
```

Configure H.323 VoIP connection

Next we configure call signaling and the codecs:

```
SN(cfg)#gateway h323 h323
SN(gw-h323)[h323]#no ras
SN(gw-h323)[h323]#faststart
SN(gw-h323)[h323]#codec g711alaw64k
SN(gw-h323)[h323]#codec g723_6k3
SN(gw-h323)[h323]#codec g729
SN(gw-h323)[h323]#exit
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface h323 CompanyOfficeA
SN(if-h323)[Company~]#codec g711alaw64k
SN(if-h323)[Company~]#exit
SN(ctx-cs)[switch]#
SN(ctx-cs)[switch]#interface h323 CompanyOfficeB
SN(if-h323)[Company~]#codec g711alaw64k
SN(if-h323)[Company~]#exit
SN(ctx-cs)[switch]#
SN(ctx-cs)[switch]#exit
SN(cfg)#
```

In addition we have to use the VoIP profile by the gateway:

```
SN(cfg)#gateway h323 h323
SN(gw-h323)[h323]#use voip-profile H323VoIPProfile
SN(gw-h323)[h323]#exit
SN(cfg)#
```

Activate CS context configuration

Prior to activating our configuration we use two 'show' commands to display part of our configuration:

```
SN(cfg)#show context cs config
Following session-router configuration sets are available:
  switch
    Interfaces:
      CompanyOfficeB
      CompanyOfficeA
      PBX
      PublicPSTN
    Routing tables:
      CalledNumberRouting
SN(cfg)#
SN(cfg)#show gateway h323 config
CURRENT H.323 GATEWAY CONFIGURATION
  State : enabled
  RAS : disabled
  Call-signaling port : 1720
  Q.931-tunneling : disabled
  Faststart : enabled
  Codecs :
    G.711 A-law, rxlen:10, txlen:10
    G.711 U-law, rxlen:20, txlen:10
    G.723, rxlen:10, txlen:10
    G.729, rxlen:10, txlen:10
  H.323-ID :
  E.164 alias :
  Binding: ip-context ’router’, interface ’eth00’ ip-address ’147.86.130.1’
SN(cfg)#
```

Finally we bind the PSTN interfaces to the ports and create simultaneously the line hunt groups and activate the gateway and CS context:

```
SN(ctx-cs)[switch]#interface pstn PBX
SN(if-pstn)[PBX]#bind port 2 0
SN(if-pstn)[PBX]#bind port 2 1
SN(if-pstn)[PBX]#exit
SN(ctx-cs)[switch]#interface pstn PublicPSTN
SN(if-pstn)[PublicP~]#bind port 2 2
SN(if-pstn)[PublicP~]#bind port 2 3
SN(if-pstn)[PublicP~]#exit
SN(ctx-cs)[switch]#exit
SN(cfg)#gateway h323
SN(gw-h323)[gw_name]#no shutdown
SN(gw-h323)[gw_name]#exit
SN(cfg)#debug session-router 5
SN(cfg)#context cs
SN(ctx-cs)[switch]#no shutdown
SN(ctx-cs)[switch]#02:47:59  SR    > Loading interfaces...
02:47:59  SR    > Resolving interface references interfaces...
```

```
02:47:59  SR    > Classifier is resolving interface references...
02:47:59  SR    > Loading session router tables...
02:47:59  SR    > Resolving routing table references within routing tables...
02:47:59  SR    > Resolving interface references within routing tables...
02:47:59  SR    > Resolving routing table references within interfaces...
02:47:59  SR    > Classifier is resolving sessionrouter references...
02:47:59  SR    > Loading and linking complete...
02:47:59  SR    > Following voice interfaces have been loaded:
02:47:59  SR    >   callapp
02:47:59  SR    >   CompanyOfficeB
02:47:59  SR    >   CompanyOfficeA
02:47:59  SR    >   PBX
02:47:59  SR    >   PublicPSTN
02:47:59  SR    > Following routing tables have been loaded:
02:47:59  SR    >   CalledNumberRouting
02:47:59  SR    > Following functions have been loaded:
02:47:59  SR    > Following number replacement tables have been loaded:

SN(ctx-cs)[switch]#
```

Show the running configuration

The configuration script for our application looks as follows:

```
  cli version 2.00

system
  clock-source 2 3
  synchronize-to-isdn-time

context ip router

interface eth00
  ipaddress 147.86.130.1 255.255.255.0
  mtu 1500

interface eth01
  ipaddress 172.16.1.1 255.255.0.0
  mtu 1500

context ip router

context cs switch
  no number-prefix national
  no number-prefix international
  called-party CalledNumberRouting 1 dest-interface CompanyOfficeA
  called-party CalledNumberRouting 2 dest-interface CompanyOfficeB
  called-party CalledNumberRouting 5 dest-interface PBX
  called-party CalledNumberRouting default dest-interface PublicPSTN
  use tone-set-profile default

interface pstn PBX
  routing dest-table CalledNumberRouting
  fallback dest-interface PublicPSTN
  bind port 2 0
```

```
    bind port 2 1

  interface pstn PublicPSTN
    routing dest-table CalledNumberRouting
    bind port 0 1

  interface h323 CompanyOfficeA
    routing dest-table CalledNumberRouting
    remoteip 146.86.130.11
    codec g711alaw64k

  interface h323 CompanyOfficeB
    routing dest-table CalledNumberRouting
    remoteip 146.86.130.24
    codec g711alaw64k

  context cs switch
    no shutdown

  profile voip H323VoIPProfile
    dtmf-relay

  gateway isoip
    shutdown

  gateway h323
    codec g711alaw64k 10 10
    codec g723_6k3 10 10
    codec g729 10 10
    faststart
    no ras
    bind interface eth00 router
    use voip-profile H323VoIPProfile
    no shutdown

  port ethernet 0 0
    medium 10 half
    encapsulation ip
    bind interface eth00 router
    no shutdown

  port ethernet 0 1
    medium 10 half
    encapsulation ip
    bind interface eth01 router
    no shutdown

  port isdn 2 0
    down
    channel-range 0 1
    l2proto pp
    l3proto pss1
    max-channels 2
    uni-side net
    up
```

```
port isdn 2 1
  down
  channel-range 0 1
  l2proto pp
  l3proto pss1
  max-channels 2
  uni-side net
  up

port isdn 2 2
  down
  channel-range 0 1
  l2proto pp
  l3proto dss1
  max-channels 2
  uni-side usr
  up

port isdn 2 3
  down
  channel-range 0 1
  l2proto pp
  l3proto dss1
  max-channels 2
  uni-side usr
  up
```

Chapter 24 CS interface configuration

Chapter contents

Introduction

This chapter provides an overview of interfaces in the CS context and describes the tasks involved in configuring them.

Within the CS context of SmartWare, an interface is a logical entity providing call routing for incoming and outgoing calls to and from ISDN 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 Session 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 56 illustrates the function of the CS interfaces. The types of CS interfaces are:

- PSTN interfaces (ISDN and POTS)
- VoIP interfaces, such as:
  - H.323 interface
  - ISoIP interface

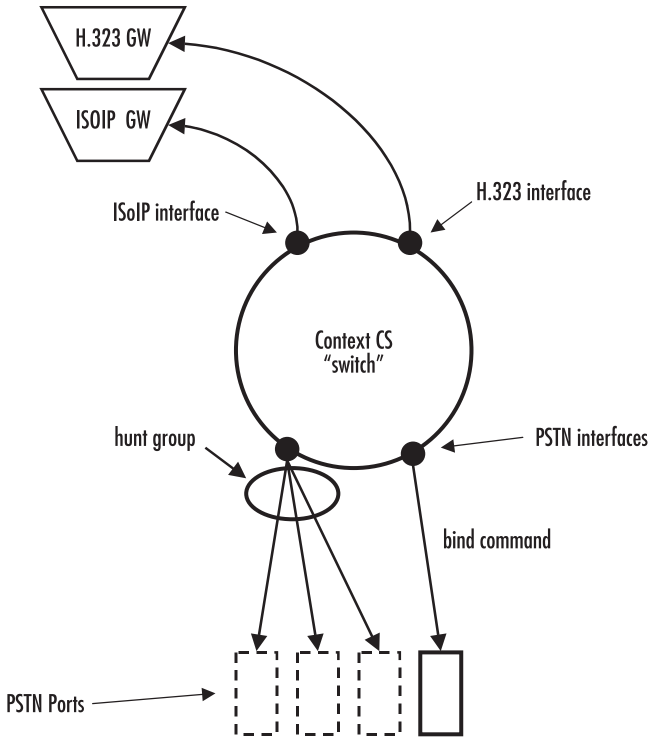


Figure 56. CS interfaces on the CS context

A PSTN interface is a CS interface type bound to a physical ISDN or POTS port. If more than one port is bound to a PSTN interface, a line hunt group is created automatically.

H.323 and ISoIP interfaces are CS interface types that provides voice over IP settings in addition to the general CS interface parameters. All H.323 and ISoIP interface on the CS context are implicitly bound to the H.323 or ISoIP gateway through SmartWare.

CS interface configuration task list

Some parameters depend upon the interface type. If it is not specifically stated otherwise, the configuration task is valid for all interfaces. Some parameters are generally set in other CS components than CS interfaces, but can be overwritten in CS interfaces. This is not described in this chapter, but in chapter 20, “DHCP configuration” on page 235, and chapter 30, “VoIP profile configuration” on page 377. To create and configure CS interfaces you have to perform the configuration tasks listed below.

- Create and configure CS interfaces
- Configure call routing (see page 298)
- Configure digit collection (advanced) (see page 299)
- Configure direct call signaling on VoIP interfaces (see page 300)
- Specify the port address on VoIP interfaces (advanced) (see page 301)
- Bind PSTN interfaces to PSTN ports and create line hunt groups (see page 303)

Create and configure CS interfaces

To configure CS interfaces you have first to enter in the CS context mode. Once entered you can create and configure your required interface by entering the CS interface configuration mode and specify the parameters. Each interface has a name. The name can be any arbitrary string of not more than 25 characters. Use a name describing the purpose of the interface as used 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 table below.

Note An interface is not defined and can not be displayed until a parameter in the interface itself is set.

This procedure describes how to create and configure CS interfaces.

Mode: Configure

Step	Command	Purpose
1	***node***(**cfg)#context cs**	Enter the CS Context Configuration Mode.
2	***node***(**ctx-cs)[switch]#interface** *if-type if-name*	Enter the CS Interface Configuration Mode & select the CS interface with type *if-type* and name *if-name* for configuration.
3	***node***(**if-type)[***if-name***]#...**	Perform the configuration tasks to configure the CS interface.
4	***node***(**if-type)[***if-name***]#exit**	Go back to the CS Context Configuration Mode
5		Repeat steps 1 to 4 to create and configure your CS interfaces

Step	Command	Purpose
6	***node***(**ctx-cs)[***switch***]#show context cs config** or ***node***(**ctx-cs)[***switch***]#interface** *if-type <?>*	Display already defined CS interfaces.
7	***node***(**ctx-cs)[***switch***]#no interface** *if-type if-name*	Delete an already defined interface.

Examples: Create an CS interface 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 pstn PublicAccess
SN(if-pstn)[PublicA~]#routing dest-interface PBX
SN(if-pstn)[PublicA~]#interface pstn <?>
  <name>                        Interface name
  PBX                           Existing interface
  PublicAccess                  Existing interface
  PublicPSTN                    Existing interface
  callapp                       Existing interface
SN(if-pstn)[PublicA~]#no interface pstn PublicPSTN
SN(ctx-cs)[switch]#interface pstn <?>
  <name>                        Interface name
  PBX                           Existing interface
  PublicAccess                  Existing interface
  callapp                       Existing interface
SN(ctx-cs)[switch]#
```

Configure call routing

SmartWare offers two levels of call routing, *basic interface routing* and *advanced session routing*. Basic interface routing allows you to forward all incoming calls on a CS interface to a destination CS interface. An optional fallback interface can also be specified.

Advanced session routing allows you to route calls to all available CS interfaces, based on a number of criteria such as calling number, destination number, and ISDN bearer capability. Further the Session Router also allows you to modify the calling and called party numbers in the call setup messages.

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 Session Router (advanced session routing).

This procedure describes how to configure basic interface routing

Mode: Context CS

Step	Command	Purpose
1	***node***(**ctx-cs)[switch]#interface** *if-type if-name*	Enter CS Interface Configuration Mode and configure interface *if-type* with name *if-name*
2	***node***(*if-type*)[*if-name*]**#routing dest-interface** *name*	Specify a destination interface for incoming calls
3	***node***(*if- type*)[*if-name*]**#fallback dest-interface** *name*	Specify a fallback interface for incoming calls
4	***node***(*if- type*)[*if-name*]**#exit**	Go back to the CS context configuration mode
5		Repeat steps 1–3 for all the required CS interfaces

Example: Configure call routing

The following example shows how to configure basic interface routing including fallback.

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface pstn PBX_trunk
SN(if-pstn)[PBX_tru~]#routing dest-interface H323_0
SN(if-pstn)[PBX_tru~]#fallback dest-interface Public_access
SN(if-pstn)[PBX_tru~]#exit
SN(ctx-cs)[switch]#
```

Configure digit collection

The SmartWare CS context supports overlap dialing on all interfaces. Some of the connected devices (PBX, ISDN network, remote gateways and gatekeepers) may however require bloc sending of the dialed number. SmartWare offers three options to collect overlap dialed digits and forward them in a single call setup message. Digit collection is applied to call setups going out from the interface to the port or gateway as illustrated in figure 57.

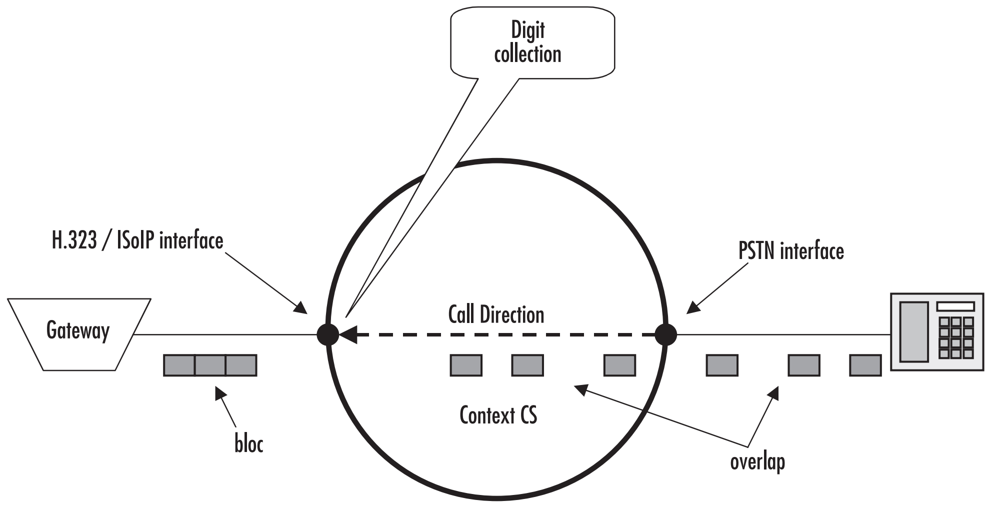


Figure 57. Digit collection in a H.323/ISoIP interface

Collected overlap dialed digits are sent providing that they comply with one or a combination of three criteria: elapse a specific timeout, receive a specific character or receive a specified number of dialed digits.

This procedure describes how to configure the digit collection

Mode: Interface below Context CS

Step	Command	Purpose
1	***node***(*if-type*)[*if-name*]**#digit-collection timeout** *seconds*	Specify the timeout to wait for the first dialed or between two dialed digits before all previous digits are sent.
2	***node***(*if-type*) [*if-name*]**#digit-collection terminating-char** *character*	Specify a character that indicates the end of the dialed digits # or *
3	***node***(*if-type*) [*if-name*]**#digit-collection nr-length** *length*	Specify a number of digits to collect

Example: Configure digit collection

In the following example the SmartNode sends the bloc of numbers either when the asterisk character '*' is received or after 4 seconds have elapsed since the last dialed digit.

```
SN(if-h323)[h323_0]#digit-collection timeout 4
SN(if-h323)[h323_0]#digit-collection terminating-char *
```

Configure direct call signaling on VoIP interfaces

Call signaling is a basic requirement needed to set up and close down a call between two endpoints. A call can be established between two VoIP interfaces, that is either between two ISoIP or between two H.323 interfaces. So that a call can be set up the VoIP interface needs the IP address of its remote device. In direct call signaling this is configured for each VoIP interface directly. In an H.323 network it is possible to obtain the remote IP addresses automatically by a gatekeeper. This is called 'gatekeeper routed call signaling'. To configure an H.323 interface for 'gatekeeper routed call signaling' refer to chapter 20, "DHCP configuration" on page 235.

Note If you specify direct call signaling for an H.323 connection make sure that the registration authentication service (RAS) is disabled. For more information on RAS refer to chapter 20, "DHCP configuration" on page 235. For an example refer to the examples at the end of this chapter.

Note Compared to H.323, the ISoIP signaling is simpler, faster and more efficient.

This procedure describes how to configure direct call signaling on a VoIP interface.

Mode: Context CS

Step	Command	Purpose
1	***node***(**ctx-cs)[switch]#interface h323** *if-name* or ***node***(**ctx-cs)[switch]#interface isoip** *if-name*	Enter interface configuration mode and configure a H.323 or ISoIP interface for 'direct call signaling'.
2	***node***(*if-type*) [*if-name*]**#remoteip** *ipaddress*	Set the remote IP address for this VoIP interface

Example: Configure direct call signaling on ISoIP interface

The following example shows how to configure 'direct call signaling' in an ISoIP interface by specifying the remote IP address

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface isoip AccessGateway
SN(if-isoip)[AccessG~]#remoteip 192.195.123.64
SN(if-isoip)[AccessG~]#
```

Specify the port address on VoIP interfaces

ISoIP and the Q.931 tunneling in H.323 can support ISDN services which use D-channel broadcast messages, such as call back and call waiting services. These features are related to physical ports on the remote gateway and not to a specific called party number.

In conventional voice circuits a D-channel message always reaches its destination, because the destination is always connected directly with a wire to the switch (e.g. PSTN). In VoIP networks a directly hard wired connection does not exist. Therefore a virtual tunnel is specified over the IP network in which the physical ports are to be virtually connected. The virtual tunnel is realized by assigning an identical port address to two VoIP interfaces. Likewise you have to configure 'direct call routing' as described above between the VoIP interfaces and the physical port. For more explanation refer to figure 58.

Note If you use this feature on a H.323 interface you have also to enable Q.931 tunneling as described in more detail in chapter 20, "DHCP configuration" on page 235.

Note If no port address is specified for an interface, all calls with no port address are routed to that interface. If a call has a port address which is not specified on a interface, the call will be rejected.

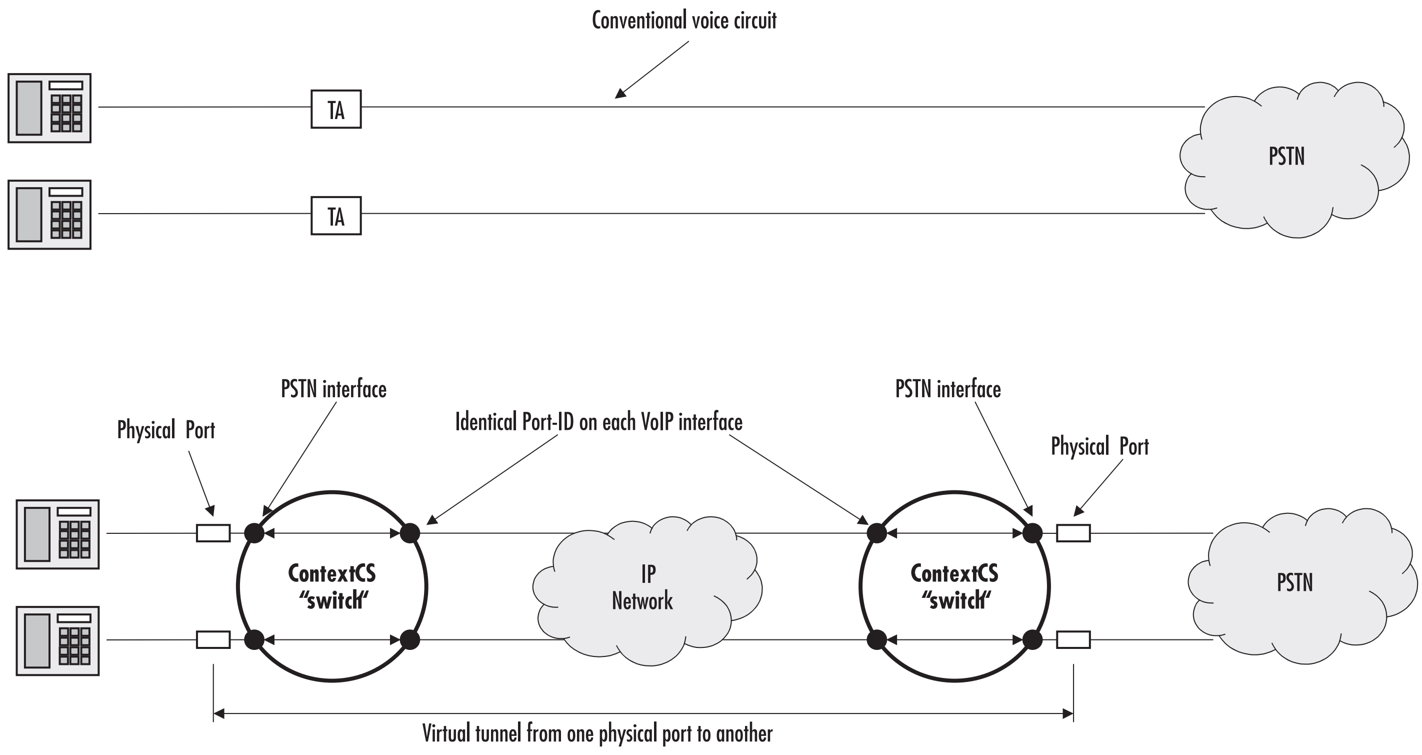

Conventional voice circuit
TA
TA
PSTN
PSTN interface
Physical Port
Identical Port-ID on each VoIP interface
PSTN interface
Physical Port
ContextCS "switch"
IP Network
ContextCS "switch"
PSTN
Virtual tunnel from one physical port to another

Figure 58. VoIP interfaces with identical port-ID determine a virtual tunnel

This procedure describes how to configure the port identifier for a H.323 or ISoIP interface

Mode: Interface below Context CS

Step	Command	Purpose
1	***node***(*if-type*)[*if-name*]**#portaddress** *address*	Specify the port address for a H.323 or ISoIP interface.

Example: Specifying a port address

The following example shows how to set the port address to value 5 for a H.323 interface and enable Q.931 tunneling to use this feature in an H.323 network.

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface h323 EnterpriseGateway
SN(if-h323)[Enterpr~]#portaddress 5
SN(if-h323)[Enterpr~]#exit
SN(ctx-cs)[switch]#exit
SN(cfg)#gateway h323 h323
SN(gw-h323)[h323]#q931-tunneling
SN(gw-h323)[h323]#
```

Bind PSTN interfaces to PSTN ports and create line hunt groups

In order to become functional the PSTN interface must be bound to the physical PSTN port from which it receives incoming calls and to which it forwards outgoing calls. Generally a PSTN interface binds only one PSTN port. If a PSTN interface binds more than one PSTN port a line hunt group is created. Hunt groups allows you to apply the PSTN interface configuration to multiple physical ports. Within the hunt groups free channels for outgoing calls are hunted on all available ports. Its are hunted in the sequence of their binding in the interface configuration.

This procedure describes how to bind the PSTN interface to a PSTN port

Mode: Interface below Context CS

Step	Command	Purpose
1	***node***(*if-type*)**[***if-name***]#bind port** *slot port*	Bind interface to a port

Example: Bind PSTN interface to PSTN ports

The following example shows how to create a line hunt group by binding multiple PSTN ports to the PSTN interface.

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface pstn PublicPSTN
SN(if-pstn)[PublicP~]#bind port 2 0
SN(if-pstn)[PublicP~]#bind port 2 1
SN(if-pstn)[PublicP~]#
```

Examples

V5 carrier access

Figure 59 on page 304 shows a V5 carrier access scenario. The association between the subscriber ISDN port and the switch port is achieved using a port address. This port address creates a virtual extension line which supports the complete ISDN services and supplementary services

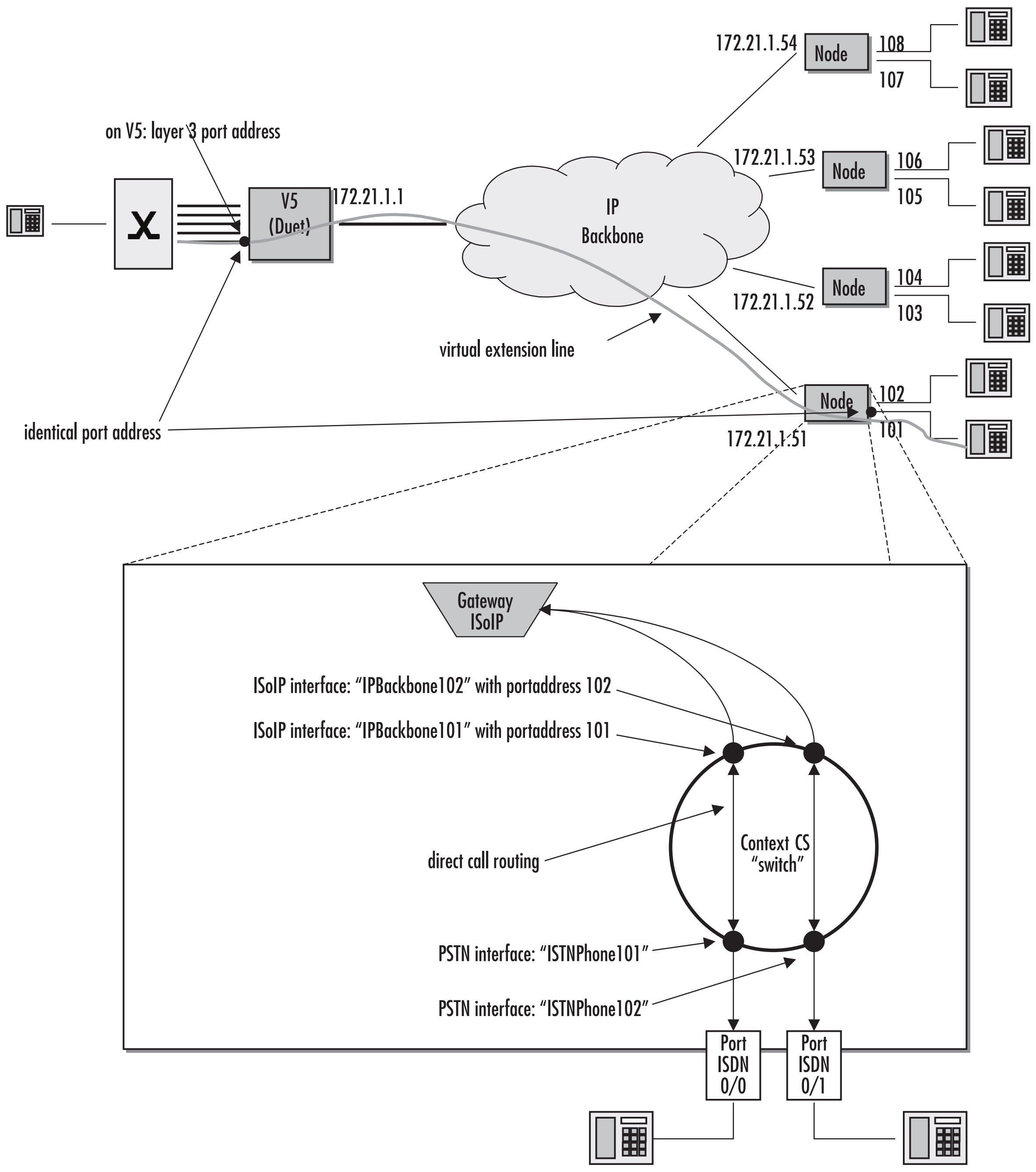


Figure 59. V5 carrier access scenario

First we configure the CS interfaces and the call routing. We need two ISoIP interfaces and two PSTN interfaces. Calls from the ISoIP interface are directly routed to the PSTN interface and vice versa:

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface isoip IpBackbone101
SN(if-isoip)[IpBackb~]#routing dest-interface ISDNPhone101
SN(if-isoip)[IpBackb~]#exit
SN(ctx-cs)[switch]#interface pstn ISDNPhone101
SN(if-pstn)[ISDNPho~]#routing dest-interface IpBackbone101
SN(if-pstn)[ISDNPho~]#exit
SN(ctx-cs)[switch]#interface isoip IpBackbone102
SN(if-isoip)[IpBackb~]#routing dest-interface ISDNPhone102
SN(if-isoip)[IpBackb~]#exit
SN(ctx-cs)[switch]#interface pstn ISDNPhone102
SN(if-pstn)[ISDNPho~]#routing dest-interface IpBackbone102
SN(if-pstn)[ISDNPho~]#exit
SN(ctx-cs)[switch]#
```

Next we specify the remote IP address for direct call signaling and set the port address:

```
SN(ctx-cs)[switch]#interface isoip IpBackbone101
SN(if-isoip)[IpBackb~]#remoteip 172.21.1.1
SN(if-isoip)[IpBackb~]#portaddress 101
SN(if-isoip)[IpBackb~]#exit
SN(ctx-cs)[switch]#interface isoip IpBackbone102
SN(if-isoip)[IpBackb~]#remoteip 172.21.1.1
SN(if-isoip)[IpBackb~]#portaddress 102
SN(if-isoip)[IpBackb~]#exit
SN(ctx-cs)[switch]#
```

Additionally we have to bind the PSTN interfaces to the PSTN ports:

```
SN(ctx-cs)[switch]#interface pstn ISDNPhone101
SN(if-pstn)[ISDNPho~]#bind port 0 0
SN(if-pstn)[ISDNPho~]#exit
SN(ctx-cs)[switch]#interface pstn ISDNPhone102
SN(if-pstn)[ISDNPho~]#bind port 0 1
SN(if-pstn)[ISDNPho~]#exit
SN(ctx-cs)[switch]#
```

Note The settings described above are normally configured together in one for an interface.

You must also configure the ISDN ports and the voice codec used. After you have finished the CS configuration you need to enable the CS context. Prior to activating the CS context we enable the debug session router monitor to display the loading of the CS context.

```
SN(ctx-cs)[switch]#debug session-router 5
SN(ctx-cs)[switch]#no shutdown
SN(ctx-cs)[switch]#
02:47:59  SR    > Loading interfaces...
02:47:59  SR    > Resolving interface references interfaces...
02:47:59  SR    > Classifier is resolving interface references...
02:47:59  SR    > Loading session router tables...
02:47:59  SR    > Resolving routing table references within routing tables...
```

```
02:47:59  SR    > Resolving interface references within routing tables...
02:47:59  SR    > Resolving routing table references within interfaces...
02:47:59  SR    > Classifier is resolving sessionrouter references...
02:47:59  SR    > Loading and linking complete...
02:47:59  SR    > Following voice interfaces have been loaded:
02:47:59  SR    >   callapp
02:47:59  SR    >   IpBackbone101
02:47:59  SR    >   IpBackbone102
02:47:59  SR    >   ISDNPhone101
02:47:59  SR    >   ISDNPhone102
02:47:59  SR    > Following routing tables have been loaded:
02:47:59  SR    > Following functions have been loaded:
02:47:59  SR    > Following number replacement tables have been loaded:
SN(ctx-cs)[switch]#
```

Q.SIG PBX networking

The example in figure 60 shows a Q.SIG PBX network scenario. To separate different tunnels for Q.SIG tunneling and to simplify the routing configuration every H.323 interface needs a unique port address.

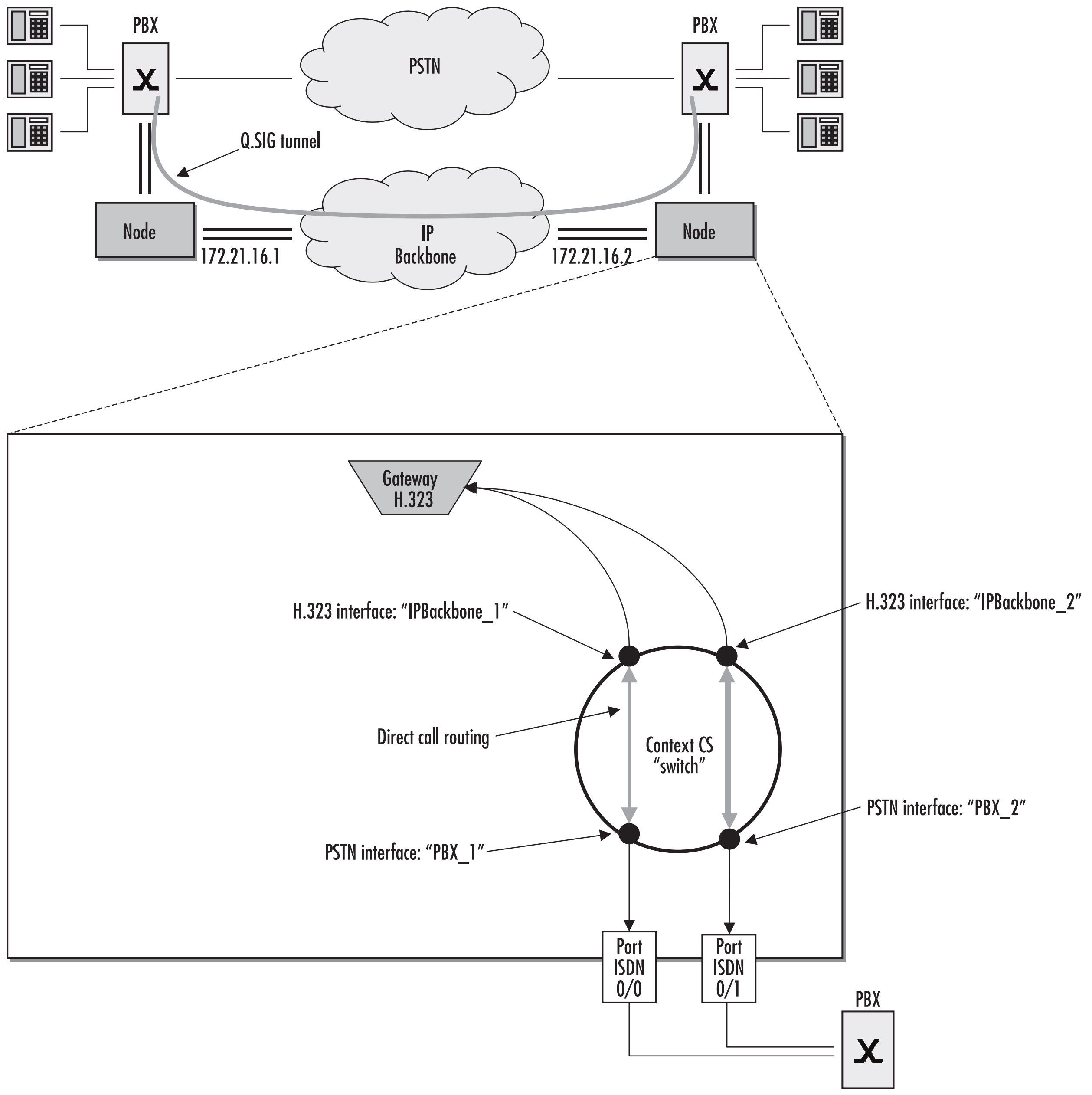


Figure 60. Q.SIG PBX network

First we configure the CS interfaces and the call routing. We need two PSTN interfaces and two H.323 interfaces. Direct call routing is used as described in the previous example:

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface h323 IPBackbone_1
SN(if-h323)[IPBackb~]#routing dest-interface PBX_1
SN(if-h323)[IPBackb~]#exit
SN(ctx-cs)[switch]#interface h323 IPBackbone_2
SN(if-h323)[IPBackb~]#routing dest-interface PBX_2
SN(if-h323)[IPBackb~]#exit
SN(ctx-cs)[switch]#interface pstn PBX_1
SN(if-pstn)[PBX_1]#routing dest-interface IPBackbone_1
SN(if-pstn)[PBX_1]#exit
SN(ctx-cs)[switch]#interface pstn PBX_2
SN(if-pstn)[PBX_2]#routing dest-interface IPBackbone_2
SN(if-pstn)[PBX_2]#exit
SN(ctx-cs)[switch]#
```

Next we configure the remote IP address for direct call signaling and specify the port addresses:

```
SN(ctx-cs)[switch]#interface h323 IPBackbone_1
SN(if-h323)[IPBackb~]#remoteip 172.21.16.1
SN(if-h323)[IPBackb~]#portaddress 1
SN(if-h323)[IPBackb~]#exit
SN(ctx-cs)[switch]#
SN(ctx-cs)[switch]#interface h323 IPBackbone_2
SN(if-h323)[IPBackb~]#remoteip 172.21.16.1
SN(if-h323)[IPBackb~]#portaddress 2
SN(if-h323)[IPBackb~]#exit
SN(ctx-cs)[switch]#
```

Because we use H.323 we have to disable the registration authentication service (RAS) and enable Q.931 tunneling. Furthermore we enable fast connect and enable the H.323 gateway with the command 'no shutdown'. These two commands are described in more detail in 20, "DHCP configuration" on page 235 on page 235.

```
SN>enable
SN#configure
SN(cfg)#gateway h323 h323
SN(gw-h323)[h323]#no ras
SN(gw-h323)[h323]#q931-tunneling
SN(gw-h323)[h323]#faststart
SN(gw-h323)[h323]#no shutdown
SN(gw-h323)[h323]#exit
SN(cfg)#
```

Next we bind the PSTN interfaces "PBX_1" and "PBX_2" to the PSTN ports:

```
SN(ctx-cs)[switch]#interface pstn PBX_1
SN(if-pstn)[PBX_1]#bind port 0 0
SN(if-pstn)[PBX_1]#exit
SN(ctx-cs)[switch]#interface pstn PBX_2
SN(if-pstn)[PBX_1]#bind port 0 1
SN(if-pstn)[PBX_1]#exit
SN(ctx-cs)[switch]#
```

In addition you have to configure the ISDN port and the voice Codec used. Refer to chapter 27, "ISDN port configuration" on page 341 or chapter 20, "DHCP configuration" on page 235 for more information on how to configure these components.

After you have finished the CS configuration you must enable the CS context. Prior to activating the CS context we enable the debug Session Router monitor to display the loading of the CS context.

```
SN(ctx-cs)[switch]#debug session-router 5
SN(ctx-cs)[switch]#no shutdown
SN(ctx-cs)[switch]#
02:47:59  SR    > Loading interfaces...
02:47:59  SR    > Resolving interface references interfaces...
02:47:59  SR    > Classifier is resolving interface references...
02:47:59  SR    > Loading session router tables...
02:47:59  SR    > Resolving routing table references within routing tables...
02:47:59  SR    > Resolving interface references within routing tables...
02:47:59  SR    > Resolving routing table references within interfaces...
02:47:59  SR    > Classifier is resolving sessionrouter references...
02:47:59  SR    > Loading and linking complete...
02:47:59  SR    > Following voice interfaces have been loaded:
02:47:59  SR    >   callapp
02:47:59  SR    >   IpBackbone_1
02:47:59  SR    >   IpBackbone_2
02:47:59  SR    >   PBX_1
02:47:59  SR    >   PBX_2
02:47:59  SR    > Following routing tables have been loaded:
02:47:59  SR    > Following functions have been loaded:
02:47:59  SR    > Following number replacement tables have been loaded:
SN(ctx-cs)[switch]#
```

Chapter 25 Session router configuration

Chapter contents

Introduction

This chapter provides an overview of Session Router tables and number manipulation functions and describes the tasks involved in configuring the Session Router in SmartWare.

There are two options for deciding where an incoming call on a CS interface is forwarded:

- Basic interface call routing: Basic interface routing can be configured directly on the CS interfaces. It is also called *direct call routing*.
- Advanced session routing: More complex call forwarding decisions can be configured in the session router

The Session Router is a very efficient and flexible tool for routing communication sessions between CS interfaces. Based on a set of routing criteria the Session Router determines the destination (interface) for every incoming call. The forwarding decisions and features are based on a set of **routing tables** and **number manipulation functions**.

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 number manipulation functions can be used to modify the calling and called party numbers according to the network requirements. Figure 61 illustrates direct call and advanced session routing. In this chapter advanced session routing is explained. For configuring direct call routing refer to chapter 24, "CS interface configuration" on page 295.

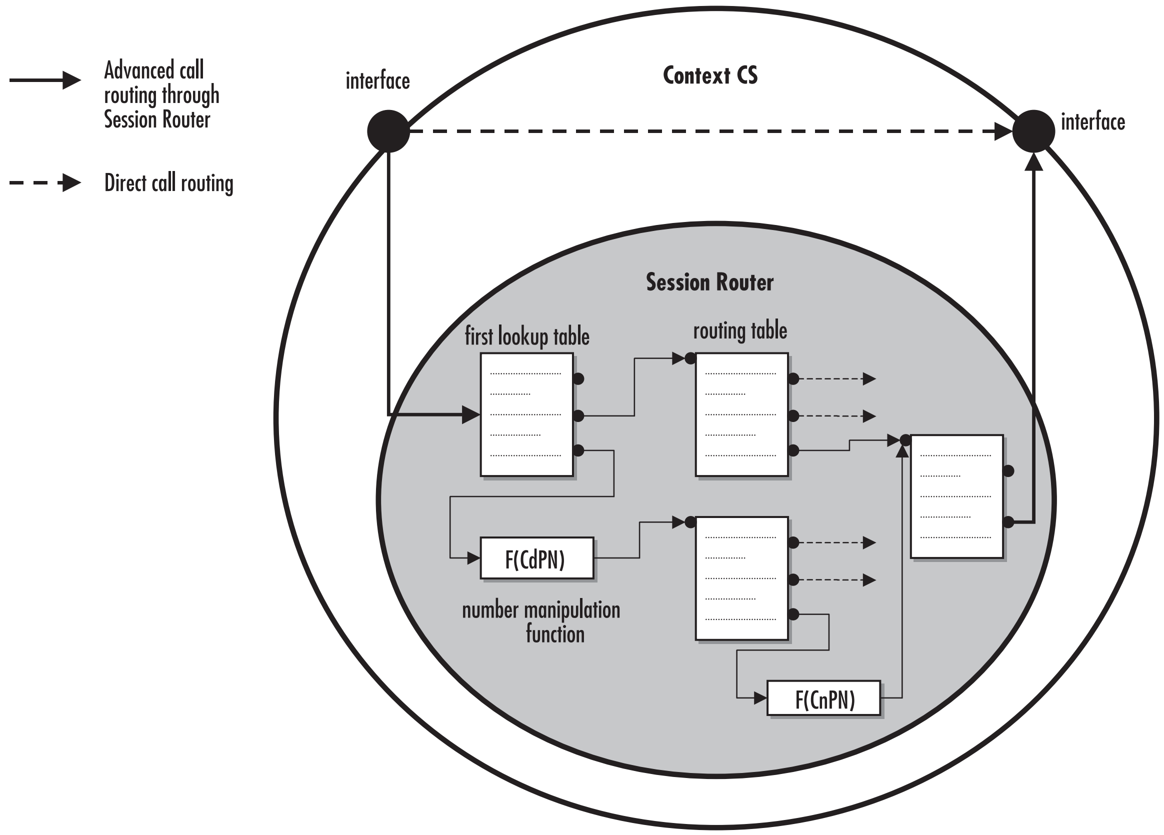


Figure 61. Direct call routing vs. advanced session routing

Due to the tree search algorithm implemented in the session router very large routing tables can be scanned very quickly with minimal impact on the call setup delay. The SmartWare session router supports the following routing criteria:

- Calling Party Number (CnPN); also called Source-Nr, A-Nr, MSN, DDI or CLIP
- Called Party Number (CdPN); also called Destination-Nr or B-Nr
- ISDN Bearer Capability; also called ISDN Service or information transfer capability (ITC)
- Day of Week; Mon–Sun
- Time of Day; hour:minute
- Date; year/month/day

Routing table structure

Every routing table has the same general format:

type <name> {<key>|default} {dest-table <dst-name> | dest-interface <dst-name> | none} [<function>]

type	Identifies the *type* of the table according to the routing criteria
<name>	Is a unique name identifying the table. Use "speaking" names
<key>	Is the matching *key*, according to the routing criteria. The Session Router reads through the table to find the best matching key. Having found one, it examines the other elements in that row. The default matching key is followed if no other row matches. If no default entry is configured and no other entry matches, the call is routed to the fallback destination configured on the interface.
dest-table	Specifies the routing table (name) to be used as the next in the decision tree. The call setup is forwarded to the table and matched against the keys.
dest-interface	Specifies the CS interface (name) to be used as the outgoing interface.
<function>	Is the name of a number manipulation function to be executed before the call setup is forwarded to the next routing table or the destination interface

Note To support broadcast features the routing table needs a default entry!

IMPORTANT

The session router allows you to solve practically any call routing and number manipulation requirement that you may have. The Session 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.

Session router configuration task list

To configure the session router, perform the tasks in the following sections and in order as listed below.

- Map out the goals for the session router (see page 314)
- Configure the entry table on circuit interfaces (see page 314)

- Configure session router tables (see page 315)
- Configure number manipulation functions (see page 320)
- Deleting routing tables and functions (see page 324)
- Activate the session router configuration (see page 325)

Map out the goals for the session router

There are many possible policies and factors that may influence the session 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), and
- Potential restrictions imposed by neighboring equipment (Gatekeepers, Remote Gateways, PBXs) on the number length or range to be used.

The session 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 tables and number manipulation functions that you need to fulfill these requirements. Finally define the decision tree (i.e. the sequence in which the tables and functions are linked together). In this 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.

Configure the entry table on circuit interfaces

To activate the advanced Session routing the first lookup table in the CS interface has to be specified as you can see in figure 61. Make sure the routing parameter on the CS interface is configured in order to forward the incoming calls to the first table of the session router.

This procedure describes how to configure the entry table in a CS interface

Mode: Context CS

Step	Command	Purpose
1	***node***(**ctx-cs**)[*switch*]**#interface** *if-type if-name*	Change to Interface Configuration Mode to specify the entry table in the interface
2	***node***(*if-type*)[*if-name*]**#routing dest-table** *name*	Specify the first lookup routing table of the Session Router.

Configure session 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.

Session router tables are created by initially configuring a first entry and then appending lines with the same table name. Refer to the individual table types detailed below on how to configure table lines.

Note The sequence of the lines is not important. The Session Router creates a search tree out of the table lines to ensure optimal search speed.

Note To remove tables the delete command can be used. Only entire tables can be deleted, not individual lines.

Broadcast handling in the session router

ISDN D-channel broadcast messages are used in various ISDN supplementary services, such as CCBS (Completion of Calls to Busy Subscriber), CW (Call Waiting), and TP (Terminal Portability). Such broadcast messages are always routed according to the default entry in the routing tables. If there is no default entry the message is dropped which will result in the loss of the corresponding call feature.

Configure number prefix for ISDN number types

The CdPN in an ISDN signaling message is 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 17 illustrates the three number types.

Table 17. ISDN number types

Type	Format Description	Example
unknown	as dialed with all leading zeros or other prefix numbers	0041 31 985xxxx
national	(area code) (local extension number)	31 985xxxx
international	(country code) (area code) (local extension number)	41 31 985xxxx

The missing prefix in the national and international number types can complicate the Session Router configuration. SmartWare therefore offers the possibility to expand these numbers before entering the first Session Router table. The configured prefix is later removed at the exit of the Session Router (i.e. when a destination interface is found).

This procedure describes how to configure number prefix

Mode: Context CS

Step	Command	Purpose
1	***node***(**ctx-cs**)[*switch*]**#** **number-prefix national** *prefix*	Adds *prefix* to all CdPNS of type national before entering the session router.
2	***node***(**ctx-cs**)[*switch*]**#** **number-prefix international** *prefix*	Adds *prefix* to all CdPNs of type international before entering the session router.

Example: Configure number prefix

```
SN[switch]#number-prefix national 0041

Input: 31985xxxx Result: 004131985xxxx

SN[switch]#number-prefix international 00

Input: 4131985xxxx Result: 004131985xxxx
```

Create a called party number routing table

The Called Party Number (CdPN) table is used to route calls based on the called party in the call set-up message. The Session Router scans the table to find the longest matching key starting with the first digit.

Note When using overlap dialing the table lookup mechanism waits for more digits until it is sure that the longest match was found.

This procedure describes how to create a CdPN routing table

Mode: Context CS

Step	Command	Purpose
1	***node***(**ctx-cs**)[*switch*]**#called-party** *tbl-name key* **dest-interface** *if-name* *or* ***node***(**ctx-cs**)[*switch*]**#called-party** *tbl-name key* **dest-table** *tbl-name*	Create a CdPN routing table *tbl-name* for destination interface *if-name* or destination table *tbl-name* by entering the first line.
2		Repeat step 1 to add lines for additional table entries.

Example: Called party number routing table

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#called-party national 001 dest-interface USVoIP-A
SN(ctx-cs)[switch]#called-party national 001320 dest-interface USVoIP-B
SN(ctx-cs)[switch]#called-party national 0044 dest-interface EuroVoIP
SN(ctx-cs)[switch]#called-party national 0049 dest-interface EuroVoIP
SN(ctx-cs)[switch]#called-party national default dest-interface DefAcc
```

Create a calling party number routing table

The Calling Party Number (CnPN) table is used to route calls based on the CnPN in the call set-up 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 Session Router looks for the longest match starting with the last digit of the calling party number.

Note The calling party number is sometimes inserted or modified by a PBX.

This procedure describes how to create a CnPN routing table

Mode: Context CS

Step	Command	Purpose
1	***node***(**ctx-cs**)[*switch*]**#calling-party** *tbl-name key* **dest-interface** *if-name* or ***node***(**ctx-cs**)[*switch*]**#calling-party** *tbl-name key* **dest-table** *tbl-name*	Create a CnPN routing table *tbl-name* with key key for destination interface *if-name* or destination table *tbl-name* by adding the first line.
2		Repeat step 1 to add lines for multiple table entries.

Example: Calling party number routing table

```
SN(ctx-cs)[switch]#calling-party exts 523 dest-interface breakout
SN(ctx-cs)[switch]#calling-party exts 525 dest-interface breakout
SN(ctx-cs)[switch]#calling-party exts 572 dest-interface DefAcc
```

Create a bearer capability routing table

The bearer capability table is used to route calls based on the bearer capability field in the ISDN setup message. This can be used to differentiate between ISDN data services and ISDN speech connections.

Note Terminals connected to analogue extensions (e.g. of a PBX) do not supply bearer capability values in their call set-up. It is therefore up to the configuration of the analogue port on the Terminal Adapter, NT or PBX to insert this value. The configuration of this value is however often omitted or wrong.

The BC value may therefore not be a reliable indication to differentiate between analogue speech, audio or Fax Group 3 connections.

The session router can route calls according to the following bearer capabilities:

- ItcAudio31: Transparent 3.1kHz channel
- ItcSpeech: Voice terminals (Telephones)
- ItcUD64: Unrestricted digital information (64 kbps)
- ItcRD64: Restricted digital information (64 kbps)

This procedure describes how to create a bearer capability routing table

Mode: Context CS

Step	Command	Purpose
1	***node***(**ctx-cs**)[*switch*]**#bearer-capability** *tbl-name key* **dest-interface** *if-name* or ***node***(**ctx-cs**)[*switch*]**# bearer-capability** *tbl-name key* **dest-table** *tbl-name*	Create a bearer-capability routing table by adding the first line.
2		Repeat step 1 to add lines for multiple table entries.

Example: Bearer capability routing table

```
SN(ctx-cs)[switch]#bearer-capability BC audio71 dest-interface local-breakout
SN(ctx-cs)[switch]#bearer-capability BC ud dest-interface ISoIP-Access
SN(ctx-cs)[switch]#bearer-capability BC video dest-interface ISoIP-Access
SN(ctx-cs)[switch]#bearer-capability BC default dest-interface VoIPCarrierA
```

Create 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.

This procedure describes how to create a time of day routing table

Mode: Context CS

Step	Command	Purpose
1	***node***(**ctx-cs**)[*switch*]**#time** *tbl-name key* **dest-interface** *if-name* or ***node***(**ctx-cs**)[*switch*]**#time** *tbl-name key* **dest-table** *tbl-name*	Create a time routing table by adding the first line.

Step	Command	Purpose
2		Repeat step 1 to add lines for multiple table entries.

Example: Time of day routing table

```
SN(ctx-cs)[switch]#time workday1 08:00-17:00 dest-table BestQuality
SN(ctx-cs)[switch]#time workday1 17:00-21:00 dest-interface VoIPCarrierA
SN(ctx-cs)[switch]#time workday1 21:00-08:00 dest-interface VoIPCarrierB
```

Create a 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 abbreviations mon; tue; wed; thu; fri; sat; and sun. To configure weekday routing table entries use the following commands starting in the CS context configuration mode.

This procedure describes how to create a day of week routing table

Mode: Context CS

Step	Command	Purpose
1	***node*(ctx-cs)[*switch*]#weekday** *tbl-name key* **dest-interface** *if-name* or ***node*(ctx-cs)[*switch*]#weekday** *tbl-name key* **dest-table** *tbl-name*	Create a day of week routing table by adding the first line.
2		Repeat step 1 to add lines for multiple table entries.

Example: Day of week routing table

```
SN(ctx-cs)[switch]#weekday DayTable1 sat dest-table LeastCost
SN(ctx-cs)[switch]#weekday DayTable1 sun dest-table LeastCost
SN(ctx-cs)[switch]#weekday DayTable1 default dest-interface VoIPCarrierA
```

Create a 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.

This procedure describes how to create a date routing table

Mode: Context CS

Step	Command	Purpose
1	***node***(**ctx-cs**)[*switch*]**#date tbl-name key dest-interface if-name** or ***node***(**ctx-cs**)[**switch**]**#date** *tbl-name key* **dest-table** *tbl-name*	Create a time routing table by adding the first line.
2		Repeat step 1 to add lines for multiple table entries.

Example: Date routing table

```
SN(ctx-cs)[switch]#date holiday2001 2001/01/01 dest-table leastCost
SN(ctx-cs)[switch]#date holiday2001 2001/01/02 dest-table leastCost
SN(ctx-cs)[switch]#date holiday2001 2001/05/01 dest-table leastCost
SN(ctx-cs)[switch]#date holiday2001 2001/12/24-2002/01/02 dest-table leastCost
SN(ctx-cs)[switch]#date holiday2001 default dest-interface VoIPCarrierA
```

Create a presentation indicator routing table

The presentation indicator defines whether the calling party number may be presented at the called party. Routing decision may be base on this information.

Procedure: To create a presentation indicator routing table

Mode: Context CS

Step	Command	Purpose
1	***node***(**ctx-cs**)[***switch***]**#presentation-indicator** *tbl-name* **{ allowed | restricted | interworking | default } dest-interface** *if-name* or ***node***(**ctx-cs**)[***switch***]**# presentation-indicator** *tbl-name* **{ allowed | restricted | interworking | default } dest-table** *tbl-name*	Create a presentation indicator routing table by adding the first line.
2		Repeat step 1 to add lines for multiple table entries.

Example: Presentation Indicator Routing Table

Reject calls that do not allow to present the calling party number:

```
SN(ctx-cs)[switch]#presentation-indicator allowed dest-interface myTerminal
```

Create a screening indicator routing table

The screening indicator provides information on the source of the Calling Party Number. Routing decisions may base on this information.

The screening indicator can have the following values:

Screening indicator	Source of the calling party number
user-not-screened	user provided, not verified
user-passed	user provided, verified and passed
user-failed	user provided, verified and failed
network	network provided

Procedure: To create a screening indicator routing table

Mode: Context CS

Step	Command	Purpose
1	***node*(ctx-cs)[*switch*]#screening-indicator** *tbl-name* **{ default | user-not-screened | user-passed | user-failed | network } dest-interface** *if-name* or ***node*(ctx-cs)[*switch*]# screening-indicator** *tbl-name* **{ default | user-not-screened | user-passed | user-failed | network } dest-table** *tbl-name*	Create a screening indicator routing table by adding the first line.
2		Repeat step 1 to add lines for multiple table entries.

Example: Screening Indicator Routing Table

Only allows calls with a verified Calling Party Number supplied by the source of the call

```
SN(ctx-cs)[switch]#screening-indicator user-passed dest-interface myTerminal
```

Configure number manipulation functions

Number manipulation functions are used to modify the call setup message and thus influence the routing decision and or the outgoing setup message leaving the Session Router. Number manipulation functions are identified by a name (string) and referenced in the routing tables.

Additionally the number type and numbering plan can be changed. Voice over IP networks may span wide areas and connect to the PSTN at different locations (international) thus imposing the need for control over the type of number and numbering plan. Especially with plain H.323 (not using Q.931 tunneling) there is no way to transport the type of number transparently.

A specific application is the 'Shared PRA' (multiple subscribers on same PRA access in Central Office Switch) which requires that all numbers have a type other than 'unknown'.

This procedure describes how to create a number manipulation function

Mode: Context CS

Table 18:

Step	Command	Purpose
1	***node*(ctx-cs)**[switch]**# [no] number-manipulation** *name* **{ cdpn | cnpn } {** **(add <param>) |** **(remove <param>) |** **(replace <param>) |** **(truncate <param>) |** **(set-presentation-indicator { allowed | restricted | interworking }) |** **(set-screening-indicator { user-not-screened | user-passed | user-failed | network}) |** **(set-type-of-number { unknown | national | international | subscriber | network specific | abbreviated }) |** **(set-numbering-plan { unknown | e164 | data | telex | national | private }) |** **convert-to-unknown |** **convert-to-specific** **}**	Create a number manipulation function identified by a "speaking" name. Specify if the function acts on the called or calling party number. Specify the action: • add digits in front of number • remove number of digits starting from the 1st digit • replace number based on translation table • truncate number to last n digits • set presentation indicator • set screening indicator • set number type • set numbering plan • convert the number type to unknown and add the configured prefix or to international and remove configured prefix if present
2		Repeat step 1 to create more than one number manipulation function

Example: Configure number manipulation functions

Add digits to CdPN: Input: 0319852525 Result: *50319852525

```
SN[switch]#number-manipulation CdPNadd0 cdpn add *5
```

Remove digits from CdPN: Input: 0319852525 Result: 9852525

```
SN[switch]#number-manipulation CdPNrm031 cdpn remove 3
```

Truncate digits of CdPN: Input: 0319852525 Result: 525

```
SN[switch]#number-manipulation CdPNtrunc3 cdpn truncate 3
```

Set number type: Input: CdPN of type 'national' Result: number of type 'international'

```
    SN[switch]#number-manipulation CdPNTypeInt cdpn set-type-of-number international
```

Set numbering plan: Input: number with numbering plan ‘e164’ Result: number with numbering plan ‘data’

```
    SN[switch]#number-manipulation CdPNNumPlanData cdpn set-numbering-plan data
```

Convert calling party number of type 'unknown' to type 'national':

Input: number (0319852525) of type 'unknown' and national prefix '0'

Result: number of type 'national' and with national prefix '0' stripped away (319852525)

Calling party numbers of type 'national' will be left untouched

```
    SN[switch]#number-prefix national 0
    SN[switch]#number-manipulation CnPNToSpecific cnpn convert-to-specific
```

Convert calling party number of type 'unknown' to type 'international':

Input: number (0041319852525) of type 'unknown' and international prefix 00

Result: number of type 'international' and with international prefix '00' stripped away (41319852525)

Calling party numbers of type 'international' will be left untouched

```
    SN[switch]#number-prefix international 00
    SN[switch]#number-manipulation CnPNToSpecific cnpn convert-to-specific
```

Convert number of type 'international' to type 'unknown':

Input: number (49319852525) of type 'international'

Result: number of type 'unknown' and with international prefix '00' (0049319852525)

Called party numbers of type 'unknown' will be left untouched

```
    SN[switch]#number-prefix international 00
    SN[switch]#number-manipulation CdPNToUnknown cdpn convert-to-unknown
```

Convert number of type 'national' to type 'unknown':

Input: number (319852525) of type 'national'

Result: number of type 'unknown' and with national prefix '0' (0319852525)

Called party numbers of type 'unknown' will be left untouched

```
    SN[switch]#number-prefix national 0
    SN[switch]#number-manipulation CdPNToUnknown cdpn convert-to- unknown
```

Create a Number Replacement Table

The replace function requires a number translation table. Translation tables are identified by a name and referenced in the Number Translation Function.

This procedure describes how to create a number replacement table

Mode: Context CS

Step	Command	Purpose
1	***node***(**ctx-cs)**[*switch*]**# translation-table** *tbl-name number-in number-out*	Create a translation table by adding the first line.
2		Repeat step 1 to add lines for multiple table entries.

Example: Create a translation table

```
SN[switch]#translation-table PISNtoDDI 550 250
SN[switch]#translation-table PISNtoDDI 551 251
SN[switch]#translation-table PISNtoDDI 552 252
SN[switch]#translation-table PISNtoDDI 553 253
```

Corresponding number manipulation function: Input: 550 Result: 250

```
SN[switch]# number-manipulation SwapDDI cdpn replace PISNtoDDI
```

Create complex number manipulation functions

Complex functions allow to combine number manipulation functions 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.

This procedure describes how to create a complex number manipulation function

Mode: Context CS

Step	Command	Purpose
1	***node***(**ctx-cs)[switch]# complex-function** *name functionname*	Create a complex function by adding the first function
2		Repeat step 1 to add additional functions to this complex function

Example: Create a complex number manipulation function

```
SN[switch]#complex-function CarriertoLocal truncate3CnPN
SN[switch]#complex-function CarriertoLocal DDItoPISN
```

Deleting Routing tables and functions

To remove individual routing tables and functions you can use the no form of the configuration command. Alternatively there is a delete command that allows to delete the complete Session Router configuration or all routing tables, translation tables or number manipulation functions.

Note Using the no form of a route table configuration command will delete the entire table not just an individual line.

This procedure describes how to delete several routing tables and number manipulation functions

Mode: Context CS

Step	Command	Purpose
1	***node***(**ctx-cs)[*switch*]#delete routing-table** *tablename*	Delete a specific routing table
2	***node***(**ctx-cs)[*switch*]#delete function** *functionname*	Delete a specific number manipulation function
3	***node***(**ctx-cs)[*switch*]#delete all-routing-tables**	Delete all routing tables in the CS context configuration
4	***node***(**ctx-cs)[*switch*]#delete all**	Delete all routing tables, functions and interfaces in the CS context configuration

Activate the session router configuration

Prior to activate the session router configuration you can show the whole context CS configuration and the entire session routing tables.

The Session 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, but changes will not be active immediately. SmartWare offers a number of possibilities to monitor and debug the CS context and Session Router configurations.

Note It is not necessary to shutdown the CS context prior to making any configuration changes.

This procedure describes how to show and activate the session router configuration

Mode: Context CS

Step	Command	Purpose
1	***node***(**ctx-cs)[*switch*]#show context cs config [*level*]**	Show the actual CS context configuration
2	***node***(**ctx-cs)[*switch*]#show running-config**	Show the whole running config includes the session routing tables
3	***node***(**ctx-cs)[*switch*]#debug session-router**	Enable the Session Router debug monitor
4	***node***(**ctx-cs)[*switch*]#no shutdown**	(Re-) activate the whole CS context configuration including the Session Router configuration.

Example

Enterprise network with local breakout and IP carrier access

Consider the following enterprise network.

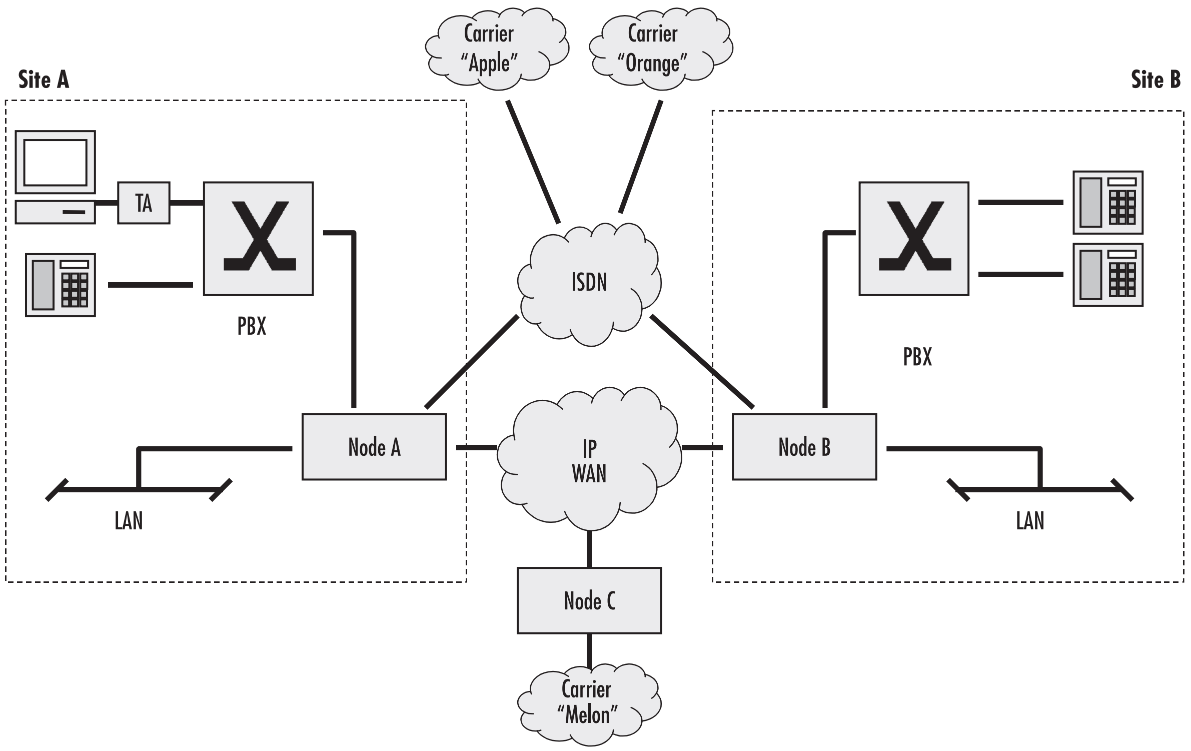


Figure 62. Session Routing Example Network

Note The SmartNodes in this Network may be owned and operated by the Company or by a Service Provider.

Two sites A and B 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 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 Session Router can be summarized as:

- Route ISDN data calls to the local breakout
- Route inter-site calls to the opposite SmartNode (Node A to Node B and vice versa)
- Route international calls to Carrier “Melon”
- Provide a fallback for all VoIP calls on the local breakout
- Route local calls to the local breakout
- Route national calls to carrier “Orange”
- Route mobile calls to carrier “Apple”
- Calls from “local_ba”, “node_b” and “node_c” are forwarded directly to “pbx_a”

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 simple configuration.

It is a good idea to specify the required Session Router elements and names before starting the configuration. A sketch may be helpful:

- Bearer Capability table named "ISDNservice", needed for requirement 1.
- Called Party Number table named "dest_a", needed for requirement 2, 3, 6 and 7
- CAC insertion for Apple "CACapple", needed to add Carrier Access Code for "Apple"
- CAC insertion for Orange "CACorange", needed to add Carrier Access Code for "Orange"
- CLI replacement for Melon "CLImelon", needed to add Carrier Access Code for "Melon"
- PSTN interfaces "pbx_a" and "local_ba", needed for requirement 4, 5 and 8.
- H.323 interface "node_b", needed for requirement 2.
- ISoIP interface "node_c", needed for requirement 3.

Figure 63 shows the corresponding CS Context and Session Router elements in Node A:

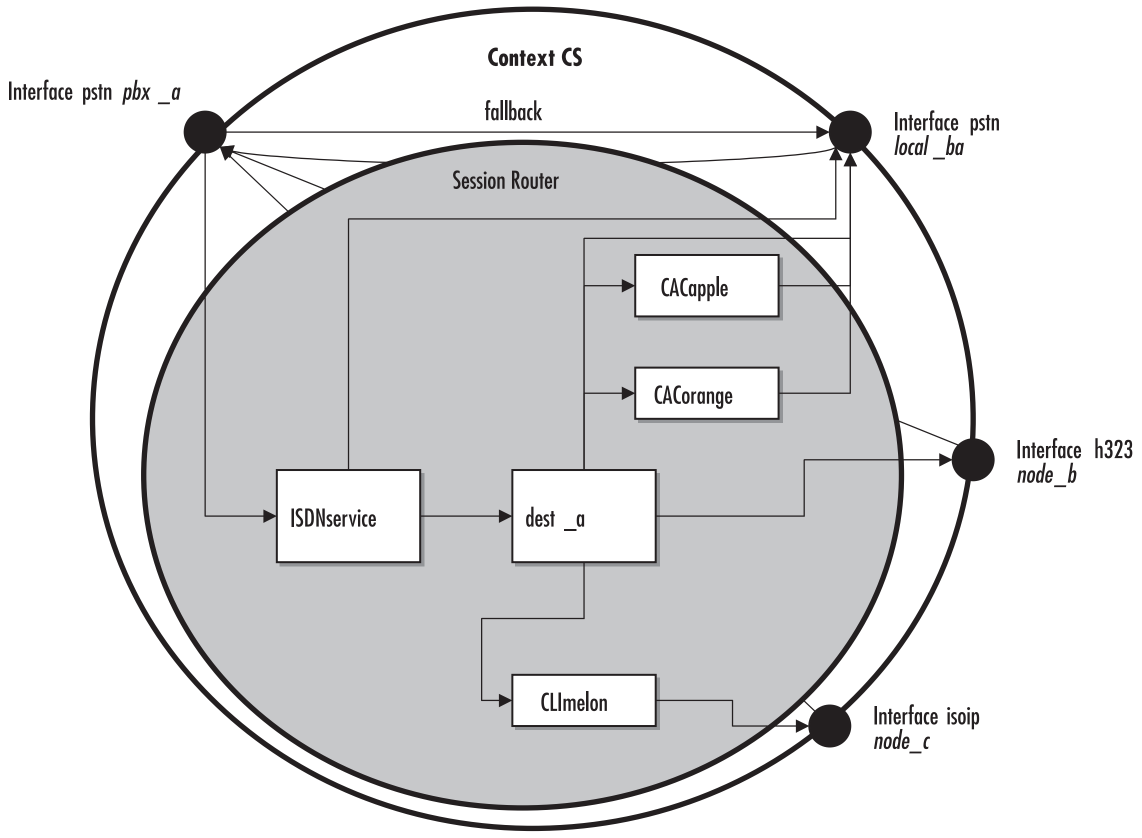


Figure 63. CS context and session router elements

We assume that the CS interfaces have already been created and configured. So we can start directly with the session 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.

```
#------------------------------------------------------------
# Session Router Config File
#------------------------------------------------------------
context cs switch
#
# Bearer capability routing table  ISDNservice
#
bearer-capability ISDNservice ud dest-interface local_ba
bearer-capability ISDNservice default dest-table dest_a
#
# Called party number routing table  dest_a
#
called-party dest_a 0 dest-interface local_ba CACorange
called-party dest_a 00 dest-interface node_c CLImelon
called-party dest_a 074 dest-interface local_ba CACapple
called-party dest_a 075 dest-interface local_ba CACapple
called-party dest_a 076 dest-interface local_ba CACapple
```

```
called-party dest_a 0336652 dest-interface node_b
called-party dest_a default dest-interface local_ba
#
# Number manipulation  CACapple
#
number-manipulation CACapple cdpn add 1055
#
# Number manipulation  CACorange
#
number-manipulation CACorange cdpn add 1066
#
# Number manipulation  CLImelon
# Truncate CLI to last 2 digits and add 0800 base number in front
#
number-manipulation CnPNtrunc3 cnpn truncate 3
number-manipulation CnPNadd_base cnpn add 08004455
#
complex-function CLImelon CnPNtrunc3
complex-function CLImelon CnPNadd_base
```

Prior to downloading this file you should make sure there are no other tables and functions in the Session Router. Duplicate entries will cause an error message.

```
SN(ctx-cs)[switch]#delete all-routing-tables
SN(ctx-cs)[switch]#delete all-functions
SN(ctx-cs)[switch]#copy tftp://172.16.36.20/configs/SRconf.cgf running-config
Download...100%
```

Now the routing elements on the CS interfaces must be configured to point to the correct tables and destination interfaces.

```
SN(ctx-cs)[switch]#interface pstn local_ba
SN(if-pstn)[local_ba]#routing dest-interface pbx_a
SN(if-pstn)[local_ba]#exit
SN(ctx-cs)[switch]#interface pstn pbx_a
SN(if-pstn)[pbx_a]#routing dest-table ISDNservice
SN(if-pstn)[pbx_a]#fallback dest-interface local_ba
SN(if-pstn)[pbx_a]#exit
SN(ctx-cs)[switch]#interface h323 node_b
SN(if-h323)[node_b]#routing dest-interface pbx_a
SN(if-h323)[node_b]#exit
SN(ctx-cs)[switch]#interface isoip node_c
SN(if-isoip)[node_c]#routing dest-interface pbx_a
SN(if-isoip)[node_c]#exit
```

The configuration is now complete. Prior to activating the configuration enable the Session Router debug monitor to check the loading of the Session Router elements.

```
SN(ctx-cs)[switch]#debug session-router
SN(ctx-cs)[switch]##context cs
SN(ctx-cs)[switch]#no shutdown
SN(ctx-cs)[switch]#17:48:54  SR    > Loading interfaces...
17:48:54  SR    > Resolving interface references interfaces...
```

```
17:48:54  SR    > Classifier is resolving interface references...
17:48:54  SR    > Loading session router tables...
17:48:54  SR    > Resolving routing table references within routing tables...
17:48:54  SR    > Resolving interface references within routing tables...
17:48:54  SR    > Resolving routing table references within interfaces...
17:48:54  SR    > Classifier is resolving sessionrouter references...
17:48:54  SR    > Loading and linking complete...
17:48:54  SR    > Following voice interfaces have been loaded:
17:48:54  SR    >   callapp
17:48:54  SR    >   local_ba
17:48:54  SR    >   pbx_a
17:48:54  SR    >   node_c
17:48:54  SR    >   node_b
17:48:54  SR    > Following routing tables have been loaded:
17:48:54  SR    >   ISDNservice
17:48:54  SR    >   dest_a
17:48:54  SR    > Following functions have been loaded:
17:48:54  SR    >   CLImelon
17:48:54  SR    >   CACapple
17:48:54  SR    >   CACorange
17:48:54  SR    >   CnPNadd_base
17:48:54  SR    >   CnPNtrunc3
17:48:54  SR    > Following number replacement tables have been loaded:

SN25(ctx-cs)[switch]#
```

Chapter 26 Tone configuration

Chapter contents

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 related to any "event" that can occur during a call, for example a call waiting indication etc. The In-band tones are referred to as "call-progress-tones". Call-progress-tones could be defined for different use, 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) 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:

- **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.
- **Alerting tone**—Tone you hear when the called party number is complete and the remote extension is ringing.
- **Busy tone**—Tone you hear when the remote extension is busy.

The three 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 a CS context and applies to all PSTN interfaces on the CS context. If it is required to have different tones for individual PSTN interfaces, the tone-set profile could be assigned directly to a PSTN interface. Doing so will override the configuration in the CS context.

Figure 64 illustrates the relation ship between call-progress-tone profiles, tone-set profiles, CS context and CS interfaces.

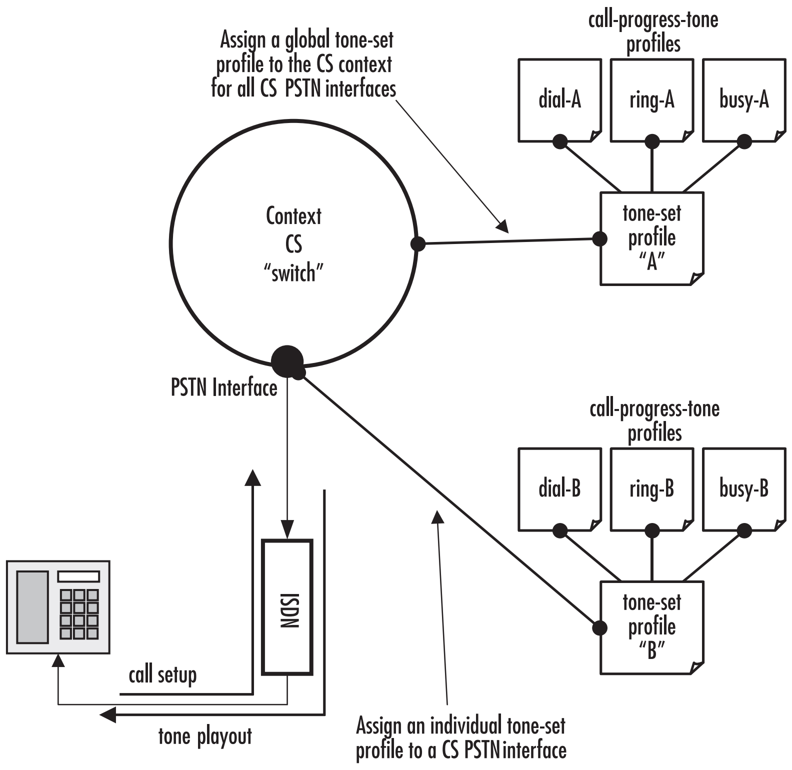

Assign a global tone-set profile to the CS context for all CS PSTN interfaces
call-progress-tone profiles
dial-A
ring-A
busy-A
Context CS "switch"
tone-set profile "A"
PSTN Interface
call-progress-tone profiles
dial-B
ring-B
busy-B
ISDN
tone-set profile "B"
call setup
tone playout
Assign an individual tone-set profile to a CS PSTN interface

Figure 64. Assign tone-sets to CS context and CS interfaces

Note There is a default tone-set named 'default', which maps the three Swiss standard in-band tones. Create a tone-set profile only if this default profile corresponds not with your country.

Tone configuration task list

To configure call progress tones, perform the tasks described in the following sections.

- Configure call-progress-tone profiles
- Configure tone-set profiles (see page 335)
- Enable generation of local in-band tones (see page 335)
- Show call-progress-tone and tone-set profiles (see page 336)

Configure 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.

This procedure describes how to configure a tone-set profile

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#profile call-progress-tone** *name*	Creates a call-progress-tone profile with *name* name and enters call-progress-tone configuration mode.
2	***node*(pf-callp)[***name***]#play** *duration frequency1 level1 [frequency2 level2]*	Defines a tone with duration *duration*, frequency *frequency1* and volume *level1*. If a second frequency is defined both frequencies are played in parallel and for the same duration
3	***node*(pf-callp)[***name***]#no play** *duration*	Defines a pause of *duration* 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 recreating 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 belgienSpec
SN(pf-callp)[belgien~]#play 330 950 -4
SN(pf-callp)[belgien~]#play 330 1400 -4
SN(pf-callp)[belgien~]#play 330 1800 -4
SN(pf-callp)[belgien~]#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 currently used configuration, 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.

This procedure describes how to configure a tone-set profile

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#profile tone-set** *name*	Creates tone-set *name* and enters tone-set profile configuration mode.
2	***node*(pf-tones)[name]#map call_progress_tone {dialtone|alertingtone|busytone}** *call-progress-tone*	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)[swiss]#map call_progress_tone dialtone dialUK
SN(pf-tones)[swiss]#map call_progress_tone alertingtone ringUK
SN(pf-tones)[swiss]#map call_progress_tone busytone busyUK
```

Enable generation of local in-band tones

The SmartNode will always generate the tones locally (as programmed in the tone-set profile) if in-band tone generation is enabled (command 'local-inband-tones') irrespective of the presence or absence of in-band tones from the IP network.

If local in-band tones are disabled (command 'no local-inband-tones') tones are played nevertheless in the following situations:

- Dial tone:The dial tone is played towards the calling party if the SessionRouter requires more digits to determine the outgoing interface
- Alerting tone:The alerting tone is played towards the calling-party if faststart is not enabled
- Busy tone:The busy tone is played towards the calling-party if Q.931-tunneling is not enabled

For enabling generation of local in-band tones first you have to assign a tone-set profile to the CS context or a CS interface, and second enable local in-band tone generation with the 'local-inband-tones' command.

Note The profile used in the CS context applies to all PSTN interfaces on the CS context. If you want to override this tone configuration on individual interfaces proceed as follows.

This procedure describes how to assign a tone-set profile to the CS context and to a CS interface

Mode: Context cs

Step	Command	Purpose
1	***node*(ctx-cs)[*switch*]#use tone-set-profile** *name*	Assign the tone-set profile to the CS context.
2	***node*(ctx-cs)[*switch*]#interface** *if-type if-name*	Enter CS interface configuration mode.
3	***node*(*if-type*)[*if-name*]#use tone-set-profile name**	Assign an other tone-set profile to a CS interface.
4	***node*(*if-type*)[*if-name*]#system**	Enter system configuration mode
5	***node*(sys)#local-inband-tones**	Enable generation of local in-band tones

Example: Assign tone-set profiles to CS context and CS interface and enable local in-band tone generation

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]#use tone-set-profile SWISS
SN(ctx-cs)[switch]#interface pstn bri0
SN(if-pstn)[bri0@switch]#use tone-set-profile USA
SN(if-pstn)[bri0@switch]#system
SN(sys)#local-inband-tones
```

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.

This procedure describes how to show call-progress-tone profiles

Mode: administrator exec

Step	Command	Purpose
1	***node*#show profile call-progress-tone [*name*]**	Display all call-progress-tone profiles or a specific with name *name*

Example: Show call-progress-tone profile

The following example shows how to display the call-progress-tone profiles.

```
SN#show profile call-progress-tone
Profiles:
---------

defaultDialtone:
  Play  5000ms (425Hz at 0dB)

defaultAlertingtone:
  Play  1000ms (425Hz at -7dB)
  Pause 4000ms

defaultBusytone:
  Play  500ms (425Hz at -7dB)
  Pause 500ms

myprofile:
  Play  5000ms (425Hz at 0dB and 3000Hz at 0B)
```

This procedure describes how to show tone-set profiles

Mode: Administrator exec

Step	Command	Purpose
1	***node*#show profile tone-set** [*name*]	Display all tone-set profiles or a specific with name *name*

Example: Show **tone-set** profile

The following example shows how to display the tone-set profile.

```
SN(cfg)#show profile tone-set

Tone set default
------------------------------------------------
DTMF high frequency level: 0 dBm
DTMF low frequency level:  0 dBm
DTMF duration:             0 ms
DTMF interspace:           0 ms
------------------------------------------------
Call progress Tone mapping:
  dialtone -> defaultDialtone
  alertingtone -> defaultAlertingtone
  busytone -> defaultBusytone
```

Example

The following example shows how to configure a tone-set profile for UK and apply it to the CS context. Figure 65 illustrates the use and map commands.

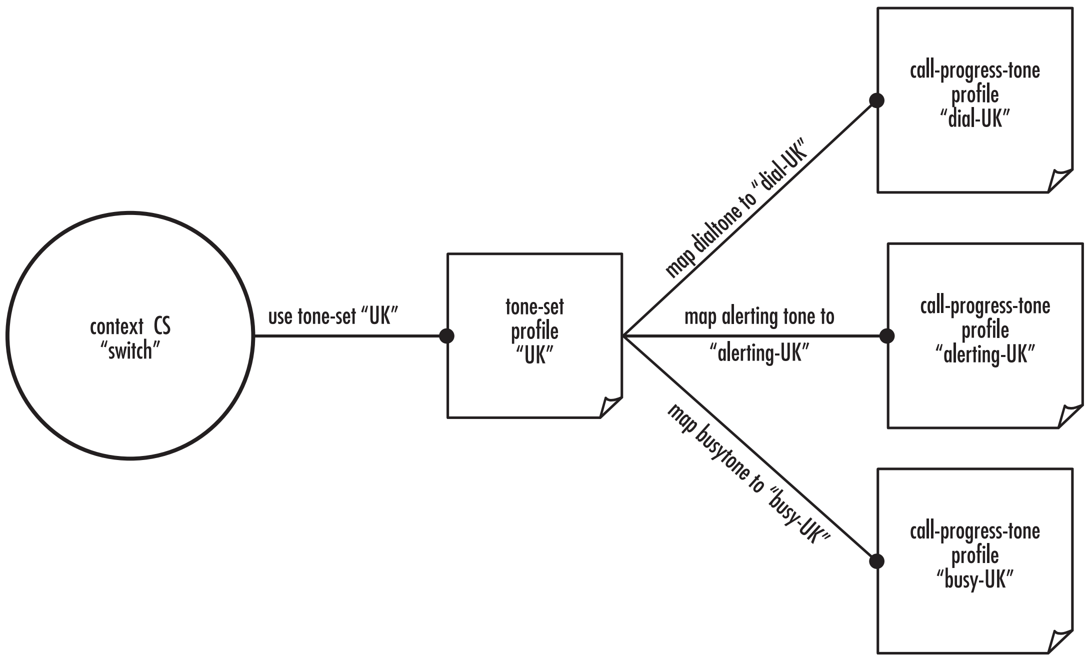

context CS
"switch"
use tone-set "UK"
tone-set profile "UK"
map dialtone to "dial-UK"
map alerting tone to "alerting-UK"
map busytone to "busy-UK"
call-progress-tone profile "dial-UK"
call-progress-tone profile "alerting-UK"
call-progress-tone profile "busy-UK"

Figure 65. Assign a tone-set profile to the CS context

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 CS context

```
SN(cfg)#context cs
SN(ctx-cs)[switch]#use tone-set-profile UK
```

Chapter 27 ISDN port configuration

Chapter contents

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 64 kbps B-channels for switched voice or data connections, one 16 kbps 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 64 kbps B-channels, one 64 kbps 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 66 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.

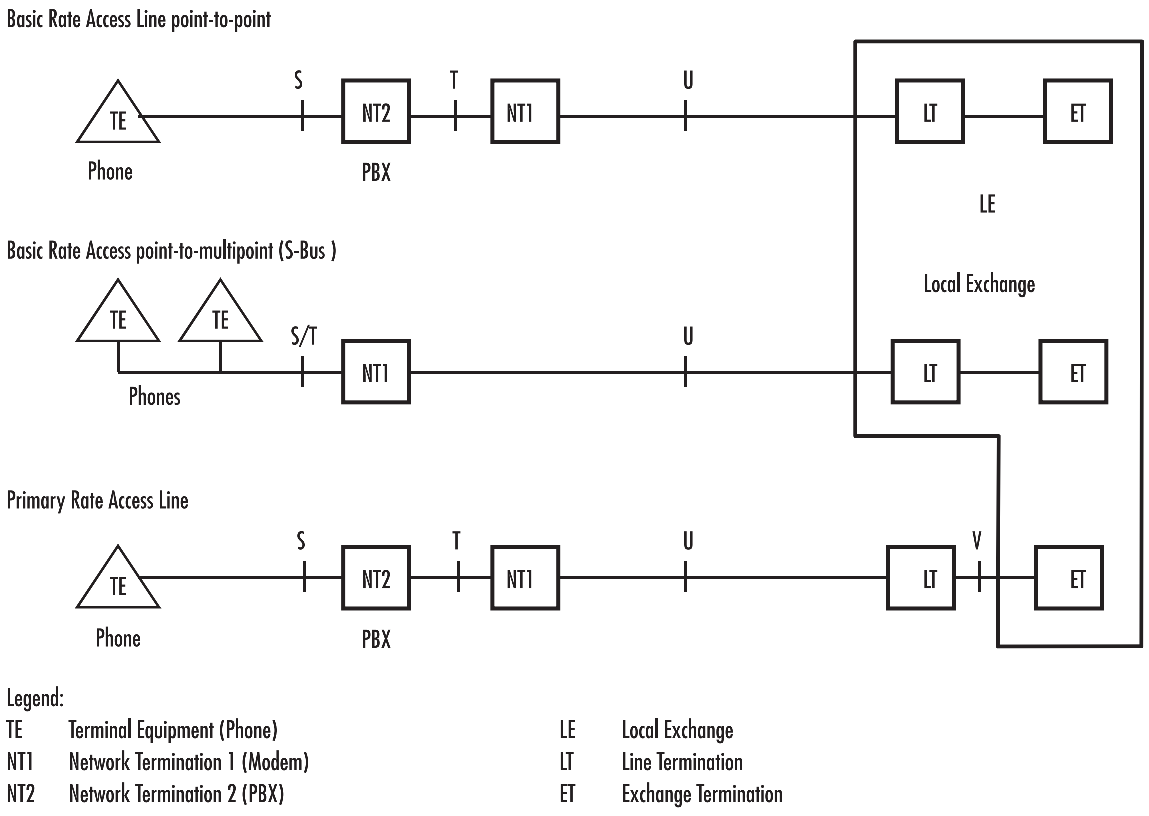

Basic Rate Access Line point-to-point
TE
Phone
S
NT2
PBX
T
NT1
U
LT
ET
LE
Local Exchange
Basic Rate Access point-to-multipoint (S-Bus)
TE
TE
Phones
S/T
NT1
U
LT
ET
Primary Rate Access Line
TE
Phone
S
NT2
PBX
T
NT1
U
LT
V
ET
Legend:
TE Terminal Equipment (Phone)
NT1 Network Termination 1 (Modem)
NT2 Network Termination 2 (PBX)
LE Local Exchange
LT Line Termination
ET Exchange Termination

Figure 66. 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.

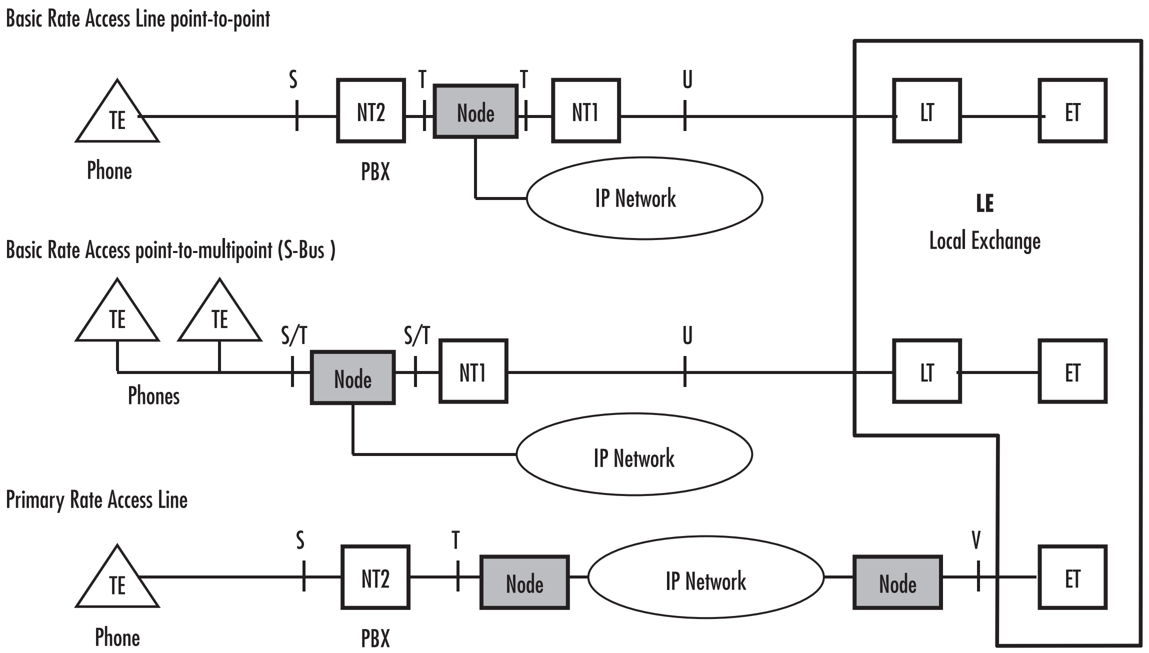
The S/T reference point is on a point-to-multipoint S-Bus. Here several terminals are connected directly to the same BRI NTU. The S and T reference points are “collapsed”. The NT2 is not represented by any equipment unit.

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 67, which illustrates some of the possible network integration options.



Legend:

TE	Terminal Equipment (Phone)	LE	Local Exchange
NT1	Network Termination 1 (Modem)	LT	Line Termination
NT2	Network Termination 2 (PBX)	ET	Exchange Termination

Figure 67. Integration in IASDN access lines

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. The 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 68.

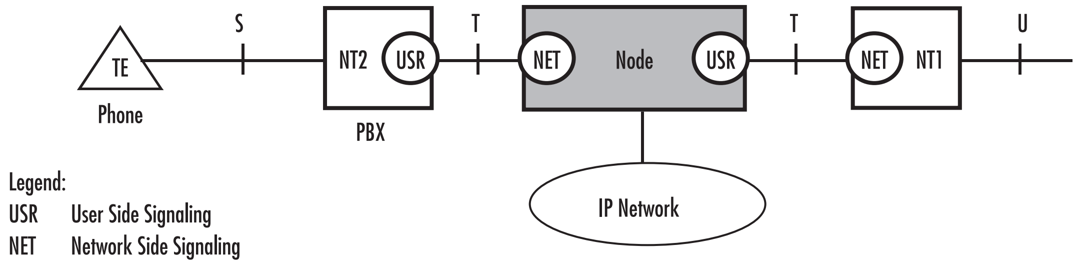


Figure 68. ISDN signaling side

IMPORTANT

Port activation deactivation—ISDN ports can only be configured when in the down state. The first configuration task explains how to disable and enable ISDN ports for configuration. When the port is down all active calls on the port are dropped. This operation must 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 64 kbps PCM channels on the ISDN ports and in the CS context. This reference clock source can be internal or 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 23, "CS context overview" on page 275 for information on configuring the reference clock.

IMPORTANT

Connector Pinout and Short circuits—Some of the SmartNode ISDN BRI ports are configurable to operate as network or terminal ports. The pinout 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 bellow to avoid miss configurations.

ISDN port configuration task list

Configuring ISDN ports typically consists of the following tasks:

- Shutdown and enable ISDN ports (see page 346)
- Configure common BRI and PRI parameters (see page 346)
- Configure BRI port parameters (see page 349)
- Configure PRI port parameters (see page 350)

Shutdown and enable ISDN ports

Prior to changing any configuration settings on an ISDN port, the port must be shut down. You can perform all configuration tasks and then activate the port, so activating all your configuration changes.

This procedure describes how to disable and activate an ISDN port

Mode: Context CS

Step	Command	Purpose
1	*node***(config)#port pstn** *slot port*	Enter CS context configuration
2	*node***(prt-pstn)[***slot/port***]#down**	Shuts the port down. All active calls are dropped!
3	Configure settings according to the sections below	
4	*node***(prt-pstn) [***slot/port***]#up**	Activates the port. At the end of the configuration procedure re-enabling the port activates the configuration changes.

Example: Shutting down and enabling an ISDN port

The following example shows how to enter the configuration mode for ISDN port 0/0, disable and activate the port.

```
172.16.40.71>enable
172.16.40.71#configure
172.16.40.71(cfg)#port pstn 0 0
172.16.40.71(prt-isdn)[0/0]#down
```

Add here the configuration changes:

```
172.16.40.71(prt-pstn)[0/0]#up
```

Configure common BRI and PRI parameters

Layer 3 signaling, the ISDN UNI side and smart-disconnect are settings applicable to both BRI and PRI ports. The several settings are described in the configuration steps below. Refer to the getting started guide included with your SmartNode device for more information on slot and port numbering conventions.

The clock source depends on the UNI-side configuration of the ISDN ports. Table 19 shows what clock-source is used in the different ISDN port configurations.

Table 19. Clock-sources used for ISDN port configurations

Port 0 UNI-side	Port 1 UNI-side	Clock Source
USR	USR	Port 1 used as reference
NET	NET	Internal clock generation
USR	NET	Port 0 used as reference (The only mode that the SmartNode 1200 supports)
NET	USR	Not supported

Note On the SmartNode 1000 series the clocking of the BRI ports is not configurable through an explicit CLI command. The SmartNode 1400 needs to be rebooted twice after changing the clock-source.

This procedure describes how to configure common BRI and PRI settings

Mode: Port ISDN

Step	Command	Purpose
1	*node***(prt-pstn)#l3proto dss1** or *node***(prt-pstn)#l3proto pss1**	Specify the ISDN layer 3 protocol. The ISDN layer 3 is the network signaling protocol. SmartWare ISDN ports support Euro-ISDN (E-DSS1) and Q.SIG (PSS1) signaling. The layer 3 signaling must correspond to the connected ISDN equipment or network.
2	*node***(prt-pstn)#uni-side usr** or *node***(prt-pstn)#uni-side net**	Specify the UNI side. This setting also specifies the master/slave setting for layer 2. NET: UNI network side and layer 2 master USR: UNI user side and layer 2 slave The layer 2 master/slave 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.

Step	Command	Purpose
3	*node***(prt-pstn)[***slot/ port***]#clock-mode master** or *node***(prt-pstn)[***slot/ port***]#clock-mode slave**	Specify the clock mode The BRI or 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 The default setting is derived from the UNI side setting as follows: • NET: clock master • UNI: clock slave (See table 20 and the **Note** below for restrictions that apply to the SmartNode 1000 Series and the Interface Card IC-4BRV)
4	*node***(prt-pstn)[***slot/ port***]#max-channels** *number*	Limits the total number of concurrent calls on the port. **Note** If the channel-range and max-channel command are used simultaneously the lower number of channel is the limiting parameter.
5 optional	*node***(prt-pstn)[***slot/ port***]#channel-hunting down** or *node***(prt-pstn)[***slot/ port***]#channel-hunting down-cyclic** or *node***(prt-pstn)[***slot/ port***]#channel-hunting up** or *node***(prt-pstn)[***slot/ port***]#channel-hunting up-cyclic**	Specify channel hunting The hunting 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.
6 optional	*node***(prt-isdn)#smart-disconnect from-isdn-calls** or *node***(prt-isdn)#smart-disconnect to-isdn-calls**	With SmartDisconnect enabled, calls are terminated immediately when a disconnect message is received. The feature can be enabled for each call direction individually. Refer to the command reference guide for details and a list of cause values.

Table 20 shows which clock-mode configurations are allowed for the SmartNode 1000 Series and which port is used as the clock source.

Table 20. Clock-Modes and clock-sources for the SmartNode 1000 Series

Port 0 clock-mode	Port 1 clock-mode	Clock Source
slave	slave	Port 1 used as reference
master	master	Internal clock generation
slave	master	Port 0 used as reference (The only configuration that SmartNode 1200 supports)
master	slave	Not supported

With the Interface Card IC-4BRV, Ports 0 and 1 can only be 'master'.

Note The SmartNode 1400 needs to be rebooted twice after changing the clock-source.

Example: Configuring 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 pstn 0 0
172.16.40.71(prt-pstn)[0/0]#l3proto dss1
172.16.40.71(prt-pstn)[0/0]#uni-side usr
172.16.40.71(prt-pstn)[0/0]#max-channels 2
```

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 pstn 0 0
172.16.40.71(prt-pstn)[0/0]#uni-side net
172.16.40.71(prt-pstn)[0/0]#clock-mode slave
172.16.40.71(prt-pstn)[0/0]#port pstn 0 1
172.16.40.71(prt-pstn)[0/1]#uni-side net
```

Configure BRI port parameters

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.

This procedure describes how to configure BRI port parameters

Mode: Port ISDN

Step	Command	Purpose
1	***node***(**prt-pstn**)[*slot/port*]**#l2proto pmp** or ***node***(**prt-pstn**)[*slot/port*]**#l2proto pp**	Specify the ISDN layer 2 protocol Make sure the connected ISDN device operates the same layer 2 protocol!

Configure PRI port parameters

Of the 32 time slots in an ISDN PRI, slot 0 is reserved for synchronization, slot 16 is used for signaling, and the remaining 30 slots can be used as B-channels for dial-up circuits. SmartWare offers various options to specify the use of these channels.

This procedure describes how to configure PRI port parameters

Mode: Port ISDN

Step	Command	Purpose
1	***node***(**prt-pstn**)[*slot/port*]**#channel-range** *min max*	Specify channel range to be used Limits the time-slots to be used for calls to the range between *min* and *max*. 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 time slots are available for use.
2	***node***(**prt-pstn**)[*slot/port*]**#channel-numbering etsi** or ***node***(**prt-pstn**)[*slot/port*]**#channel-numbering pss1-old**	Specify channel numbering 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	***node***(**prt-pstn**)[*slot/port*]**# [no] permanent-layer2**	Enables the ISDN Layer 2 permanently. By default, the ISDN Layer 2 is not enabled permanently, i.e. the first call enables it. (Currently only available with IC-T1V)

Example

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 hunting.

```
172.16.40.71(cfg)#port pstn 1 0
172.16.40.71(prt-isdn)[1/0]#clock-mode master
172.16.40.71(prt-isdn)[1/0]#channel-range 1 20
172.16.40.71(prt-isdn)[1/0]#max-channels 10
172.16.40.71(prt-isdn)[1/0]#channel-hunting down-cyclic
```

Example

Assume the scenario as illustrated in figure 69:

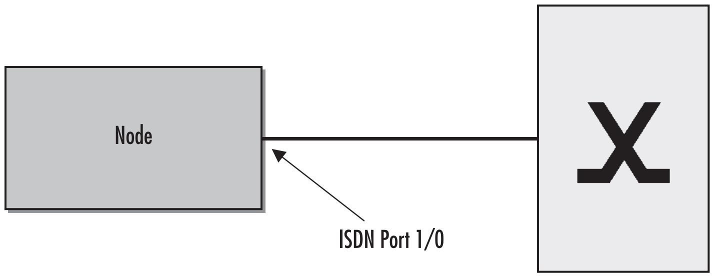


Figure 69. PBX connected to ISDN port 1/0

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 pstn 1 0
172.16.40.71(prt-isdn)[1/0]#down
172.16.40.71(prt-isdn)[1/0]#uni-side net
172.16.40.71(prt-isdn)[1/0]#clock-mode slave
172.16.40.71(prt-isdn)[1/0]#channel-numbering etsi
172.16.40.71(prt-isdn)[1/0]#max-channels 8
172.16.40.71(prt-isdn)[1/0]#up
```

Chapter 28 POTS port configuration

Chapter contents

Introduction

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

This chapter includes the following sections:

- Shutdown and enable POTS ports (see page 355)
- Configuration of common POTS port parameters (see page 356)
- POTS port configuration task list (see page 356)

POTS stands for *plain old telephone system* and is an acronym for analog telephony. 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. Inalp's IC-4ab POTS interface card for the 2000 series SmartNodes provides 4 POTS FXS ports for hooking up 4 analog phone sets (or other analog devices) to the VoIP gateway functionality of the SmartNode.

POTS signaling

The signaling procedure used on POTS 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, IC-4ab interface card etc.).

Dialing on analog phones is accomplished by means of pulse or tone dialing. With pulse dialing the line (current) is modulated in an on/off scheme according to the digit dialed; the digit 5 for instance causes 5 current interruptions on the line (0 would cause 10 of these interruptions). This kind of signaling has long been supplanted by tone (frequency) dialing and thus most analog phones in use today do not support it anymore. Therefore pulse dialing is not supported by the IC-4ab interface card.

Frequency or tone dialing uses DTMF tones to signal the digits 0 through 9, # and *. There are 4 lower and 4 upper frequencies and combining 1 frequency of each group to a dual tone pair signals a specific digit. It is this signaling scheme that is supported by the IC-4ab. If unsure whether a phone supports pulse or tone dialing (or both) simply hook such a phone to the (properly configured) IC-4ab and dial any number—pulse dialing will be audible since it is in most cases done with relays.

IMPORTANT

Port activation deactivation—POTS ports can only be configured when in the down state. The first configuration task explains how to disable and enable POTS ports for configuration. When the port is down all active calls on the port are dropped. This operation must only be performed during planned down times.

IMPORTANT

Connector pinout and short-circuits—The ports of the IC-4ab use pins 2 and 3 on an RJ-11 jack.

IMPORTANT **Available codecs for IC-4ab**—Only G.711 and G.723 or G.729 are supported on the IC-4ab, that is it is not possible to use G.723 on one port of the IC-4ab and G.729 at the same time on another port.

Shutdown and enable POTS ports

Prior to changing any configuration settings on a POTS port, the port must be shut down. You can perform all configuration tasks and then activate the port, so activating all your configuration changes.

This procedure describes how to disable and activate an POTS port.

Mode: Context CS

Step	Command	Purpose
1	**(config)#port pstn** *slot port*	Enter POTS port configuration mode
2	**(prt-pstn)[***slot/port***]#down**	Shutdown the port. All active calls are dropped!
3		Configure settings according to the sections below
4	**(prt-pstn) [***slot/port***]#up**	Activate the port. At the end of the configuration procedure re-enabling the port activates the configuration changes.

Example: Shutting down and enabling a POTS port.

The following example shows how to enter the configuration mode for POTS port 0/0, disable and activate the port.

```
172.16.40.71>enable
172.16.40.71#configure
172.16.40.71(cfg)#port pstn 1 0
172.16.40.71(prt-pstn)[1/0]#down
```

Add here the configuration changes:

```
172.16.40.71(prt-pstn)[1/0]#up
```

Configure common POTS port parameters

Some parameters pertain to all POTS ports available on an IC-4FXS or a SmartNode 4000 and are configured as shown below.

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 must be downloaded to the IC-4FXS prior operation of a POTS phone. 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 file for your country is available.

Procedure: Configure common POTS port parameters

Mode: Port PSTN

	Command	Purpose
Step 1	***node*(config)#port pstn** *slot* **all**	Enter configuration mode for all POTS ports
Step 2	***node*(prt-pstn)[*slot/port*]#use profile pots** *profile*	Select a profile containing the country specific settings of the port attributes (ring voltage etc.). The country settings are stored in a file which comes together with the software for the IC-4FXS. The available country profiles are listed when entering this command. The names listed are composed as follows: < ISO3166-1-Alpha-2 2 digit country code> Examples (currently available profiles): • 'ch' for Switzerland • 'us' for the United States of America / Canada • 'gb' for Great Britan • 'etsi' for Europe (ETSI Standard 'EuroPOTS' The default profile (not displayed in the list when entering the command) is Switzerland.

Configure POTS port parameter

This section describes the steps for the configuration of a specific POTS port.

Procedure: Configure the POTS port parameters

Mode: Configure

	Command	Purpose
Step 1	***node*(config)#port pstn** *slot port*	Enter POTS port configuration mode
Step 2 optional	***node*(prt-pstn)[*slot/port*]# subscriber-number** *number*	Configure the subscriber number (MSN in ISDN terms) of this analog phone

	Command	Purpose
Step 3 optional	*node*(prt-pstn)[*slot/port*]#caller-id 1 [format { bell | etsi }] or *node*(prt-pstn)[*slot/port*]#caller-id pre-ring [format { bell | etsi }]	Caller-id (calling party number) is sent to the phone (display) after first ringing. Caller-id is sent to the phone before ringing starts. Format can be either ETSI or Bellcore. In Europe use ETSI, in the U.S./Canada use Bellcore,
Step 4 optional	*node*(prt-pstn) [*slot/port*]# end-of-call-signaling { 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).
Step 5 optional	*node*(prt-pstn)[*slot/port*]#meter-pulses	Enable the sending of meter pulses (tax pulses) to the phone attached to this port

Chapter 29 Gateway configuration

Chapter contents

Introduction

This chapter provides an overview of ISoIP and H.323 gateways and describes the tasks involved in configuring them.

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 there is an ISoIP and an H.323 gateway. The ISoIP and H.323 interfaces in the CS context are implicitly bound to these gateways. The H.323 gateway must be bound explicitly to interfaces in the IP context. The ISoIP gateway detects the correct IP interface on the IP context for its call automatically, therefore no binding is needed. Figure 70 illustrates the function of the gateways. SmartWare currently supports one instance of each gateway. The name of the H.323 gateway is *h323* and the one for the ISoIP gateway is *isoip*.

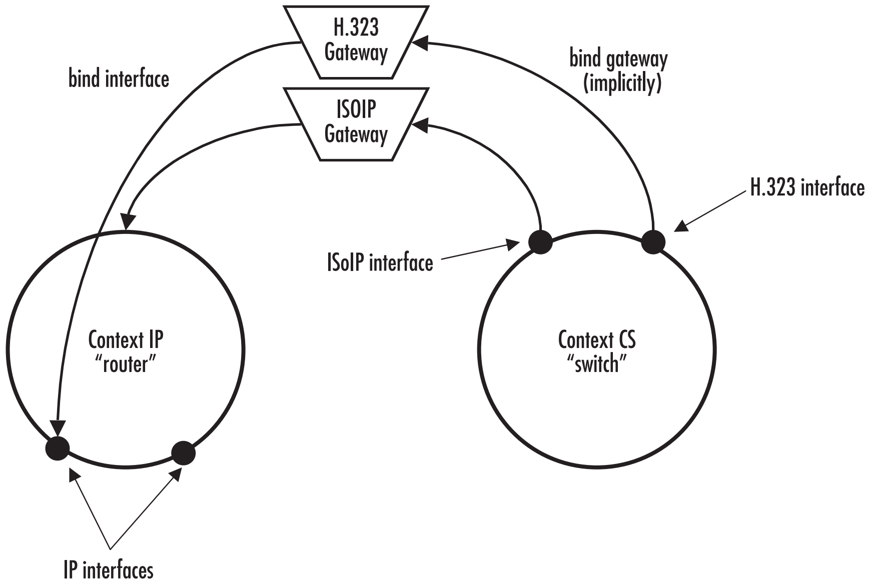


Figure 70. Gateways between IP and CS contexts

Gateway configuration task list

This chapter describes the configuration of the H.323 and ISoIP gateways. Some parameters could be configured in the gateway configuration mode and overwritten in another configuration mode. For example the default value for the voice codec for ISoIP connections is set in the gateway configuration mode and could be overwritten in the ISoIP interface. *All* possible configurations which are involved in a specific configuration topic are described in the respective configuration task. There are some differences between the ISoIP and H.323 gateway. If it is not otherwise stated, the configuration task is valid for both gateways.

- Configure codec selection and fast connect
- Configure registration authentication service in an H.323 gateway (advanced) (see page 365)
- H.323 registration authentication service gatekeeper registration type (advanced) (see page 366)
- Enable Q.931 tunneling for an H.323 connection (advanced) (see page 367)
- Configure H.235 security for H.323 (see page 368)
- Show and enable the gateway configuration (see page 368)

Configure codec selection and fast connect

Introduction

Voice channels occupy 64 kbps using PCM (pulse code modulation) coding. Over the years, compression techniques have been developed allowing a reduction in the required bandwidth while preserving voice quality. Such techniques are implemented as 'codecs'. Although many proprietary compression schemes exist, most H.323 devices today use codecs that were standardized by bodies such as the ITU-T for the sake of interoperability across vendors. Different compression schemes can be compared using four parameters:

- Compressed voice rate—the codec compresses voice from 64 kbps down to a certain bit rate. Some network designs have a preference for low-bit-rate codecs. Most codecs can accommodate different target compression rates such as 8, 6.4 and even 5.3 kbps. Note that this bit rate is for audio only. When transmitting packetized voice over the network, protocol overhead (such as RTP/UDP/IP/Ethernet) is added on top of this bit rate, resulting in a higher actual data rate.
- Complexity—the higher the complexity of implementing the codec, the more CPU resources are required.
- Voice quality—compressing voice in some codecs results in very good voice quality, while others cause a significant degradation.
- Digitizing delay—Each algorithm requires that different amounts of speech are buffered prior to the compression. This delay adds to the overall end-to-end delay. A network with excessive end-to-end delay, often causes people to revert to a half-duplex conversation (“How are you today? over…”) instead of the normal full-duplex phone call.

Which codec you can use depends on the VoIP gateway. In ISoIP there are more codecs available than in H.323. Codecs can be set in the VoIP gateways as well as in the CS interfaces.

When the codec is configured, you may also set the packet size. In SmartWare it is possible to specify the packet size of the transmitted voice packets. Larger packet sizes significantly reduce the overall bandwidth but add to the packetization delay as the sender needs to wait longer to fill up the payload. In contrast to broadcast-type media transmission (e.g. RealAudio), a two-way phone conversation is sensitive to latency.

Most callers notice round-trip delays when they exceed 250 ms, so the one-way latency budget would typically be 150 ms. Beyond that round-trip latency, callers start feeling uneasy when holding a two-way conversation and usually end up talking over each other. At 500 ms round-trip delays and beyond, phone calls are impractical. For reference, the typical delay when speaking through a geo-stationary satellite is 150–500 ms. For this reason be sure you understand the effects of changing the packet size before you change it. The default value is usually suitable so that you don't have to specify the packet length.

Note There is no *right* CODEC. The choice of what compression scheme to use depends on what parameters are more important for a specific installation. In practice, G.723 and G.729 are more popular than G.726 and G.728. For an overview of used codecs in SmartWare see appendix D, "Used IP ports & available voice codecs in SmartWare" on page 445.

Configure used codec for an ISoIP connection

The codec used for an ISoIP connection could be set in the ISoIP gateway as well as in the CS interface as follows. In the ISoIP gateway a default codec is set. If you do not specify a codec in the ISoIP interface, the default codec specified in the ISoIP gateway will be used. Otherwise the codec specified in the ISoIP gateway will be replaced by the codec specified in the ISoIP interface. In the same way it is possible to specify the packet size of the transmitted voice packets.

This procedure describes how to configure the used codec for an ISoIP connection

Mode: Gateway ISoIP

Step	Command	Purpose
1	***node*(gw-isoip)[isoip]#codec** *<tab>*	List all available codecs on the ISoIP gateway
2	***node*(gw-isoip)[isoip]#codec** *codec [packet-length]*	Specify the default codec and the packet size of the transmitted voice packet for all calls over the ISoIP gateway.
3	***node*(gw-isoip)[isoip]#exit**	Optional: Only if necessary: Change to interface configuration mode to overwrite the default codec for an interface
4	***node*(cfg)#context cs**	
5	***node*(ctx-cs)[switch]#interface isoip** *if-name*	
6	***node*(if-isoip)[if-name]#codec** *<tab>*	List all available codecs on the ISoIP interface
7	***node*(if-isoip)[if-name]#codec codec [packet-length]**	Overwrite the default codec specified in the ISoIP gateway for this ISoIP interface with codec *codec*.

Example: Selecting an ISoIP codec

The following example shows how to use the g711Alaw64k (64 kbps) codec as default voice codec for the ISoIP gateway with 20 ms for the length of transmitted RTP packets.

```
SN>enable
SN#configure
SN(cfg) #gateway isoip ISOIP
SN(gw-isoip)[isoip]#codec g711alaw64k 20
```

Note It is recommended not to use the codec *Netcoder* because of its inferior voice quality. The codec *Netcoder* does not work with SmartNode 1200 at all.

Configure used codec for an H.323 connection and enable fast connect

The codec used for an H.323 connection could be set in the H.323 gateway as well as in the H.323 interface as follows. In the H.323 gateway a list of all possible codecs is defined. In the H.323 interface one codec of those listed in the H.323 gateway must be specified. The actual codec used for the H.323 connection depends on whether fast connect is used or not as listed in table 21.

Table 21. Codec Selection in H.323

	A side (outgoing)	B side (incoming)
With fast connect	**1.** If present SmartWare sends codec list in gateway **2.** If present SmartWare sends codec specified in interface as first codec (preferred) plus the codecs specified in the gateway **3.** If present with tag 'exclusive' SmartWare sends only this codec. **Note** The codecs specified in the interfaces must also be present in the gateway codec list.	**1.** SmartWare selects the first codec from the incoming codec list which is present in codec list in gateway; the Session Router then selects the interface which contains a codec that matches the selected codec; the selected codec is used for the bearer channel. **2.** If no interface matches incoming codec, SmartWare chooses the interface with no codec specified; the selected codec is nevertheless used for the bearer channel. **3.** If no interface matches incoming codec and if no interface is present with no codec specified the call is rejected.
Without fast connect	**1.** SmartWare sends codec list in gateway; codecs specified in interface (preferred or exclusive) are without effect.	**1.** SmartWare selects the first codec from the incoming codec list which is present in codec list in gateway; SmartWare chooses interface with no codec specified. **2.** If no interface is present with no codec specified the call is rejected.

Similarly for an H.323 connection it is possible to specify the packet size of the transmitted voice packets, and additionally announce length capability for received voice packets to the remote VoIP device.

A *regular* call setup with H.323 requires about 10 TCP segments or more to be transmitted, because several parameters are negotiated, for example the codec. Because a normal call setup is often too slow, a fast connect is possible.

This procedure describes how to configure the used codec on an H.323 connection and enable fast connect

Mode: Gateway H.323

Step	Command	Purpose
1	***node***(gw-h323)[h323]#codec *<tab>*	List all available codecs on the H.323 gateway
2	***node***(gw-h323)[h323]#codec codec [tx-packet-length] [rx-packet-length]	Specify the default codec, transmitted packet size and length capability for received packets for all calls over the H.323 gateway.
3		Repeat step 2 to define all possible codecs used on this H.323 gateway
4	***node***(gw-h323)[h323]#faststart	Enable fast connect for a H.323 call set-up
5	***node***(gw-h323)[h323]#exit	Change to interface configuration mode to select one of the listed codecs in the H.323 gateway for the H.323 interface
6	***node***(cfg)#context cs	
7	***node***(ctx-cs)[switch]#interface h323 *if-name*	
8	***node***(if-h323)[if-name]#codec *codec* [exclusive]	Define the preferred codec on a fast connect call set up. It is not possible to redefine the packet lengths. If the codec is specified as exclusive, only this codec can be used for the connection

Example: Selecting an H.323 codec

The following example shows how to define a list of codecs in the H.323 gateway with 20 ms for the length of transmitted RTP packets and 40 ms for the announced length capability for received RTP packets. Further use of codec G.711 on the H.323 interface is specified. Moreover fast connect is enabled for the H.323 call set-ups.

```
SN>enable
SN#configure
SN(cfg)#gateway h323 h323
SN(gw-h323)[h323]#codec g711alaw64k 20 40
SN(gw-h323)[h323]#codec g723_6k3 20 40
SN(gw-h323)[h323]#codec g729 20 40
SN(gw-h323)[h323]#faststart
SN(gw-h323)[h323]#exit
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface h323 H323
SN(if-h323)[H323]#codec g711alaw64k
SN(if-h323)[H323]#
```

Note There is more than one possibility to establish a H.323 connection faster. Fast-start is one of them. Another is to exchange H.245 messages soon (before the CONNECT message of the H.323 call signaling). This feature is enabled per default. If some compatibility problems pass, this feature could be disabled with the command 'no early-h245'.

Enable T.38 fax over IP relay

T.38 is an ITU-T protocol for the transmission of fax (T.30) over IP. In T.38 the fax signal is demodulated at the sender side and transmitted over UDP or TCP over the IP network to the receiver which in turn modulates the signal to T.30 again.

Note Fax relay with T.38 must be enabled in the VoIP profile, either the default profile or a specifically configured VoIP profile.

This procedure describes how to enable the H.323 gateway to transmit fax with the T.38 protocol

Mode: Gateway H.323

Step	Command	Purpose
1	***node*(gw-h323)[h323]#codec t38_udp**	Enables T.38 operation if the gateway detects a fax signal from one of the PSTN lines.

Configure registration authentication service in an H.323 gateway

As mentioned in previous chapters a call setup can use *direct call signaling* and *gatekeeper routed call signaling*. For gatekeeper routed call signaling the H.323 network needs a gatekeeper device. Gatekeepers manage H.323 zones, which are logical collections of devices such as all H.323 devices within an IP subnet. Gatekeepers for example provide address translation (routing) for the devices in their zone. This could be, for instance, the translation between internal and external numbering systems. Another important function for gatekeepers is providing admission control, specifying what devices can call what numbers.

To use the gatekeeper the SmartNode has to register by the gatekeeper, therefore the registration authentication service has to be enabled. The SmartNode has to register by a name or names, therefore some aliases have to be specified. Furthermore the gatekeeper discovery could be specify on automatically or manually.

Normally the remote IP address specified in the CS interface is not set if gatekeeper routed call signaling is enabled, because the gatekeeper supplies the remote destination IP address. Nevertheless the remote IP address is set in the CS interface the call over this interface goes to the specified IP address but the gatekeeper will be asked for permission.

Redundancy is of great importance in communications 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 gatekeeper addresses. These addresses are used for RAS in a round-robin fashion. Once communication with one of the gatekeepers is lost the SmartNode falls back to the next address in the list. All entries from the list can be removed by applying the no form of the command.

Note If you have to change the call signaling port use the command 'call-signaling-port' in the H.323 gateway configuration mode.

This procedure describes how to enable the registration authentication service.

Mode: Gateway H.323

Step	Command	Purpose
1	***node***(gw-h323)[h323]#gatekeeper-discovery auto [*gkid*] or ***node***(gw-h323)[h323]#gatekeeper-discovery manual ***ip-address ip-port*** *[gkid]*	Specify that the gatekeeper discovery has to be done automatically or Specify the gatekeeper for the SmartNode explicitly.
2	***node***(gw-h323)[h323]#alias h323-id *name* or ***node***(gw-h323)[h323]#alias e164 *number*	Sets the identifier(s) for registration on the gatekeeper with a string or with a phone number
3		Repeat step 2 to add more that one alias to the configuration.
4	***node***(gw-h323)[h323]#supported-prefix *prefix*	Registers a prefix with the gatekeeper. The gatekeeper will route a calls with called party numbers starting with this prefix to this SmartNode
5		Repeat step 4 to add more that one prefix to the configuration.
6	***node***(gw-h323)[h323]#ras	Enable the registration authentication service protocol for gatekeeper registration and client resolution

Example: Configuring registration authentication service

The following example shows how to configure the registration authentication service 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 Berne1
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]#1supported-prefix 03198525
SN(gw-h323)[h323]#ras
```

H.323 registration authentication service gatekeeper registration type

H.323 gatekeepers usually allow configuring the registration type (terminal or gateway) of the registrant (registration authentication service). For administrative purposes it may be desirable to change the registration type on the gatekeeper. The gateway however needs to register with the same registration type; registration type mismatch may result in registration failure.

This procedure describes how to configure the registration type of the registration with the gatekeeper

Mode: Gateway H.323

Step	Command	Purpose
1	***node***(gw-h323)[h323]#terminal-type {terminal | gateway }	Set the registration type of the gatekeeper registration. Default registration type is gateway.

Example: Configuring registration authentication service registration type

The following example shows how to configure the registration authentication service registration type.

```
SN(cfg)#gateway h323 h323
SN(cfg)#terminal-type gateway
```

Enable Q.931 tunneling for an H.323 connection

Q.931 tunneling is able to support ISDN supplementary services which use D-channel broadcast messages, such as call back and call waiting services. In ISoIP Q.931 tunneling is always enabled, in H.323 it must be enabled or disabled explicitly.

The three different tunneling options are provided in order to achieve best interoperability with certain H.323 gateways. All three tunneling options work between two SmartNode. Depending on the tunneling option used the set of services provided may differ. The tunneling option isoip-sp has some restrictions when operated solely between two SmartNodes (due to the protocols asymmetry). The default tunneling option is isoip-2. When the peers in the H.323 network are SmartNodes (no third party gateways) using the tunneling option isoip-2 will afford the best results.

Note Most gatekeepers do not support tunneled Q.931 messages. When using a gatekeeper without Q.931 tunneling support make sure that RAS is disabled (no ras).

Note Irrespective of the tunneling option (isoip-2, isoip-sp, isoip-ig) Q.931 tunneling is automatically switched off if the H.323 peer (terminal, gateway) does not support tunneling (on a per call basis). In such cases the call will be established with 'plain' H.323 (without the services afforded by tunneling).

This procedure describes how to enable Q.931 tunneling in H.323 and select the appropriate tunneling option.

Mode: Gateway H.323

Step	Command	Purpose
1	***node***(gw-h323)[h323]# [no] q931-tunneling [isoip-2 | isoip-ig]	Select the appropriate Q.931 tunneling protocol.

Example: Configuring Q.931 tunneling and tunneling option

The following example shows how to enable Q.931 tunneling and tunneling option isoip-2 on the H.323 gateway.

```
SN (cfg)#gateway h323 h323
SN (cfg)#q931-tunneling isoip-2
```

Disabling the H.225 status enquiry

Some H.323 gateways do not answer H.225 status enquiries. The H.323 gateway within sends such H.225 status enquiry messages by default in order to detect unexpected disconnects of calls. This command allows to disable sending of H.225 status enquiry messages.

Mode: Gateway H.323

Step	Command	Purpose
1	***node***(**gw-h323)[h323]#no h225-status-enquiry**	Disables sending of H.225 Status Enquiry messages.

Disabling the 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 enabled by default, but 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	***node***(**gw-h323)[h323]#no h245-tunneling**	Disables sending of H.225 Status Tunneling messages.

H.323 transmission of Q.931 presentation and screening indicators (octet 3A)

The Q.931 presentation and screening indicators (octet 3a of the calling party number in the setup message) can now be also tunneled through H.323. The following values of these indicators are provided if the incoming ISDN or H.323 channels does not provide them:

- Presentation Indicator: 'Presentation allowed'
- Screening Indicator: 'User provided, not screened'

Mode: Gateway H.323

Step	Command	Purpose
1	***node***(**gw-h323)[h323]# [no] clip-clir-support**	Enables the H.323 tunneling of the Q.931 Presentation and Screening Indicators. Default: disabled

Configure H.235 security for H.323

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 'gencryptopassword.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

	Command	Purpose
Step 1	*node*(gw-h323)[h323]#**h235security master-password** *master-password*	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.
Step 2	**C:\getcryptopassword** *h235-password master-password*	Generates H.235 password by means of the master password with the encryption tool.

	Command	Purpose
Step 3	*node*(gw-h323)[h323]#**h235security encrypted-password** *h235-password* or *node*(gw-h323)[h323]#**h235security clear-password** *h235-password*	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.
Step 4	*node*(gw-h323)[h323]#**h235security time-window** *time-window*	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.
Step 5	*node*(gw-h323)[h323]#**h235security version {v1 | v2}**	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.
Step 6	*node*(gw-h323)[h323]#**h235security sender-id** *sender-id*	Sets the ID of the Node. Must be a string containing 2, 4, 6, ... characters.
Step 7	*node*(gw-h323)[h323]#**h235security general-id** *general-id*	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.
Step 8	*node*(gw-h323)[h323]#**h235security**	If all parameters are set, enables H.235 security. Otherwise returns an error message.
Step 9	*node*(gw-h323)[h323]#**show h235security**	Shows status of H.235 security module.
Step 10	*node*(gw-h323)[h323]#**debug h235security [***debug-level***]**	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 encrypted-password
21dafa5dfc7399e5cef9cc138dabd22f
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]#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 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
```

Show and enable the gateway configuration

Finally, before enabling the gateway SmartWare must know which IP interface belongs to which gateway. In ISoIP this is detected automatically, in H.323 the gateway must be bound to the respective IP interface.

In order to become active the ISoIP or H.323 gateway must be enabled. Before enabling the H.323 gateway it is possible to display the actual configuration.

This procedure describes how to enable ISoIP or H.323 gateway

Mode: Gateway ISoIP or gateway H.323

Step	Command	Purpose
1	***node*(gw-*type*)[*gw-name*]#bind interface if-name**	Only H.323: Bind H.323 gateway to IP interface
2	***node*(gw-*type*)[*gw-name*]#show gateway h323 config**	Only H.323: Show H.323 gateway configuration
3	***node*(gw-*type*)[*gw-name*]#no shutdown**	Enable the H.323 or ISoIP gateway

Example: Enabling an H.323 gateway configuration

The following example shows how to bind to an IP interface, display and then enable an already defined H.323 gateway.

```
SN(cfg)#gateway h323 h323
SN(gw-h323)[h323]#bind interface eth00
SN(gw-h323)[h323]#show gateway h323 config
CURRENT H.323 GATEWAY CONFIGURATION
  State : enabled
  RAS : disabled
  Call-signaling port : 1720
  Q.931-tunneling : disabled
  Faststart : enabled
  Codecs :
    G.711 A-law, rxlen:10, txlen:10
    G.711 U-law, rxlen:20, txlen:10
    G.723, rxlen:10, txlen:10
    G.729, rxlen:10, txlen:10
  H.323-ID :
  E.164 alias :
  Binding: ip-context ’router’, interface ’eth00’ ip-address ’172.16.40.123’
SN(gw-h323)[h323]#no shutdown
```

Examples

Branch offices in an enterprise network

Figure 71 shows a branch office in Linn and a branch office in Zurich connected to the main office in Berne over ISoIP. Zurich and Berne are linked over an 2 Mbps direct copper DSL wire and use the voice codec G.711 for this high-rate connection. The local office in Linn is linked to Berne over an 500 kbps leased line and therefore uses the low-bit-rate voice codec G.723.

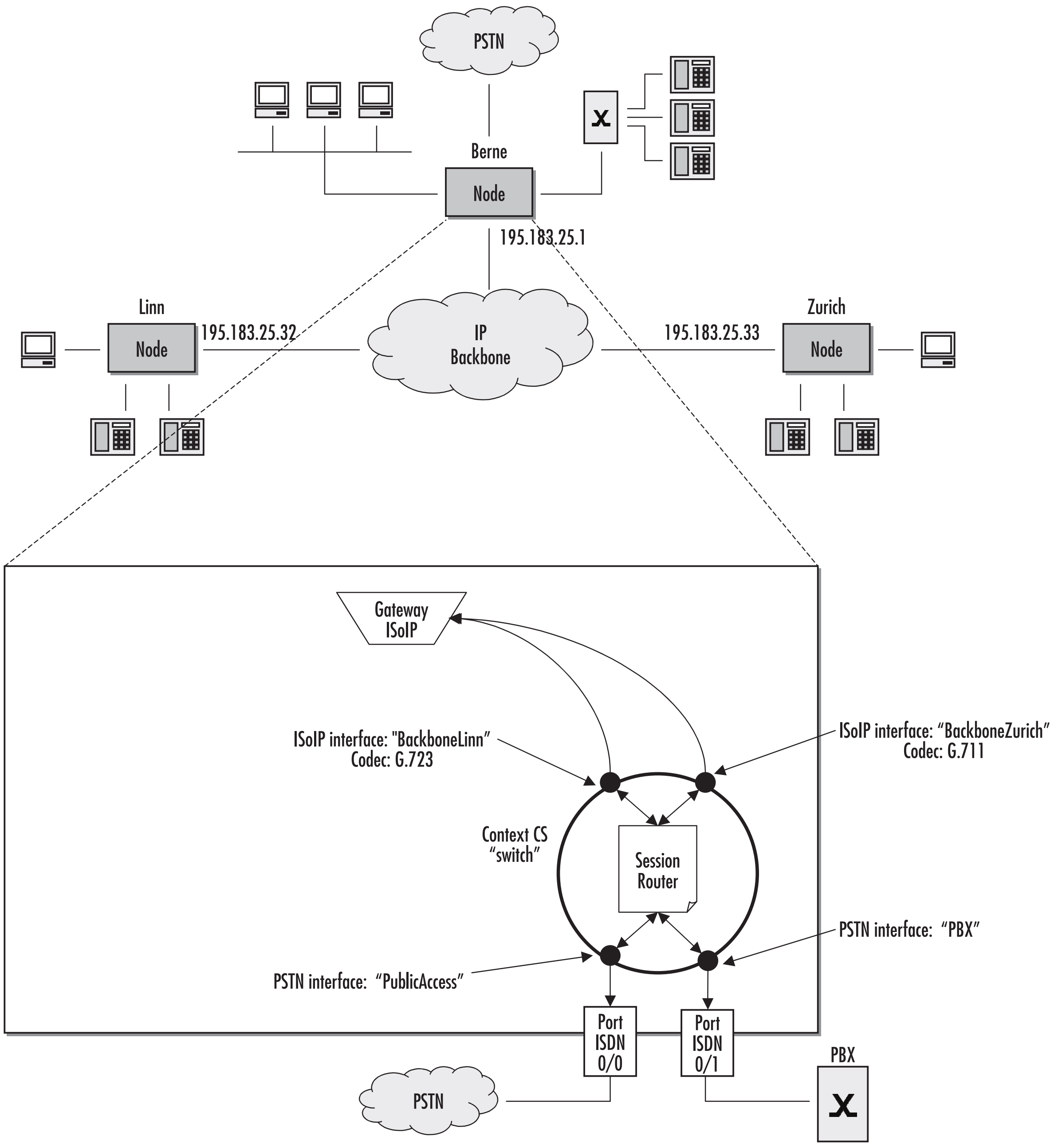


Figure 71. Branch offices in an enterprise network

First we create the CS interfaces and configure the call routing (without session router). We need two ISoIP interfaces because we have two different voice codecs from the branch offices.

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface pstn PBX
SN(if-pstn)[PBX]#routing dest-table CdPnRouting
SN(if-pstn)[PBX]#fallback dest-interface PublicAccess
SN(if-pstn)[PBX]#exit
SN(ctx-cs)[switch]#interface pstn PublicAccess
SN(if-pstn)[PublicA~]#routing dest-table CdPnRouting
SN(if-pstn)[PublicA~]#exit
SN(ctx-cs)[switch]#
SN(ctx-cs)[switch]#interface isoip BackboneLinn
SN(if-isoip)[Backbon~]#remoteip 195.183.25.32
SN(if-isoip)[Backbon~]#routing dest-table CdPnRouting
SN(if-isoip)[Backbon~]#
SN(ctx-cs)[switch]#interface isoip BackboneZurich
SN(if-isoip)[Backbon~]#remoteip 195.183.25.33
SN(if-isoip)[Backbon~]#routing dest-table CdPnRouting
SN(if-isoip)[Backbon~]#exit
SN(ctx-cs)[switch]#
```

The configuration steps for the VoIP profile and the ISDN Ports are omitted. Instead we show the configuration steps to configure the codec. In the ISoIP gateway codec G.711 is specified as the default codec. For the ISoIP interface "BackboneLinn" we have to specify codec G.723 explicitly.

```
SN(ctx-cs)[switch]#exit
SN(cfg)#gateway isoip isoip
SN(gw-isoip)[isoip]#codec g711alaw64k 20
SN(gw-isoip)[isoip]#exit
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface isoip BackboneLinn
SN(if-isoip)[Backbon~]#codec g723_5k3 60
SN(if-isoip)[Backbon~]#exit
SN(ctx-cs)[switch]#
```

Finally we enable the CS configuration by enabling the ISoIP gateway and the CS context. Note that we have omitted the binding of the CS interfaces to the ISDN ports.

```
SN(ctx-cs)[switch]#debug session-router
SN(ctx-cs)[switch]#no shutdown
SN(ctx-cs)[switch]#02:47:59  SR    > Loading interfaces...
02:47:59  SR    > Resolving interface references interfaces...
02:47:59  SR    > Classifier is resolving interface references...
02:47:59  SR    > Loading session router tables...
02:47:59  SR    > Resolving routing table references within routing tables...
02:47:59  SR    > Resolving interface references within routing tables...
02:47:59  SR    > Resolving routing table references within interfaces...
02:47:59  SR    > Classifier is resolving sessionrouter references...
02:47:59  SR    > Loading and linking complete...
```

```
02:47:59  SR    > Following voice interfaces have been loaded:
02:47:59  SR    >   callapp
02:47:59  SR    >   BackboneLinn
02:47:59  SR    >   BackboneZurich
02:47:59  SR    >   PBX
02:47:59  SR    >   PublicAccess
02:47:59  SR    > Following routing tables have been loaded:
02:47:59  SR    >   CnPNRouting
02:47:59  SR    > Following functions have been loaded:
02:47:59  SR    > Following number replacement tables have been loaded:
SN(ctx-cs)[switch]#exit
SN(cfg)#gateway isoip isoip
SN(gw-isoip)[isoip]#no shutdown
SN(gw-isoip)[isoip]#exit
SN(cfg)#
```

Gatekeeper in LAN based telephony

Figure 72 illustrates LAN based telephony with a gatekeeper. We configure the SmartNode for gatekeeper routed call signaling, and in addition we use fast connect to set up a call.

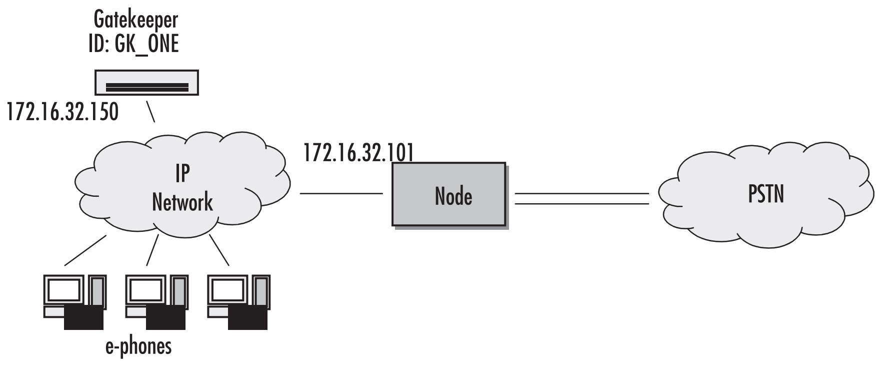


Figure 72. Gatekeeper in LAN based telephony

First we configure the CS interfaces and the call routing. We need one H.323 interface and one PSTN interface.

Note We don't specify the remote IP address on the H.323 interface.

```
SN>enable
SN#configure
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface pstn PublicAccess
SN(if-pstn)[PublicA~]#routing dest-interface H323LAN
SN(if-pstn)[PublicA~]#exit
SN(ctx-cs)[switch]#interface h323 H323LAN
SN(if-h323)[H323LAN]# routing dest-interface PublicAccess
SN(if-h323)[H323LAN]#exit
SN(ctx-cs)[switch]#
```

The configuration steps for the VoIP profile and the ISDN Ports are omitted. Next we configure the Smart-Node for 'gatekeeper routed call signaling'. Therefore we specify the alias, the gatekeeper discovery and enable the registration authentication service (RAS). Additionally we enable fast connect. Because we use H.323 we have to bind the H.323 gateway to the IP interface.

```
SN(ctx-cs)[switch]#exit
SN(cfg)#gateway h323 h323
SN(gw-h323)[h323]#alias h323-id *5
SN(gw-h323)[h323]#gatekeeper-discovery auto GK_ONE
SN(gw-h323)[h323]#ras
SN(gw-h323)[h323]#faststart
SN(gw-h323)[h323]#bind interface eth00
SN(gw-h323)[h323]#exit
SN(cfg)#
```

Finally we enable the CS configuration by enabling the ISoIP gateway and the CS context. Note that we have omitted the binding of the CS interfaces to the ISDN ports. We also display the CS configurations with the show commands.

```
SN(cfg)#show gateway h323 config
CURRENT H.323 GATEWAY CONFIGURATION
  State : enabled
  RAS : enabled, RAS UDP port : 1719
  Gatekeeper: auto-discovery (GKID:’GK_ONE’)
  Call-signaling port : 1720
  Q.931-tunneling : disabled
  Faststart : enabled
  Codecs :
    G.711 A-law, rxlen:10, txlen:10
    G.711 U-law, rxlen:20, txlen:10
    G.723, rxlen:10, txlen:10
    G.729, rxlen:10, txlen:10
  H.323-ID :
    *5
  E.164 alias :
  Binding: ip-context ’router’, interface ’eth00’ ip-address ’172.16.32.101’
SN(cfg)#
SN(cfg)#show context cs config
Following session-router configuration sets are available:
  switch
    Interfaces:
      PublicAccess
      H323LAN
    Routing tables:
SN(cfg)#
SN(cfg)#gateway h323 h323
SN(gw-h323)[h323]#bind interface eth00
SN(gw-h323)[h323]#no shutdown
SN(gw-h323)[h323]#exit
SN(cfg)#context cs
SN(ctx-cs)[switch]#no shutdown
SN(ctx-cs)[switch]#
```

Chapter 30 VoIP profile configuration

Chapter contents

Introduction

This chapter gives an overview of SmartWare VoIP profiles, how they are used and describes the tasks involved in VoIP profile configuration.

A VoIP profile summarizes the most relevant settings for VoIP connections and is assigned to the VoIP gateways H.323 or ISoIP. Each VoIP gateway must use a VoIP profile. The settings in the VoIP profile apply to all calls going through that gateway. Figure 73 illustrates the relations between VoIP profiles, gateways and CS interfaces. The configurable components are as follows:

- DTMF relay
- Echo canceller
- Silence compression and comfort noise
- Voice volume gain
- Dejitter buffer
- Post and high-pass filters
- Fax

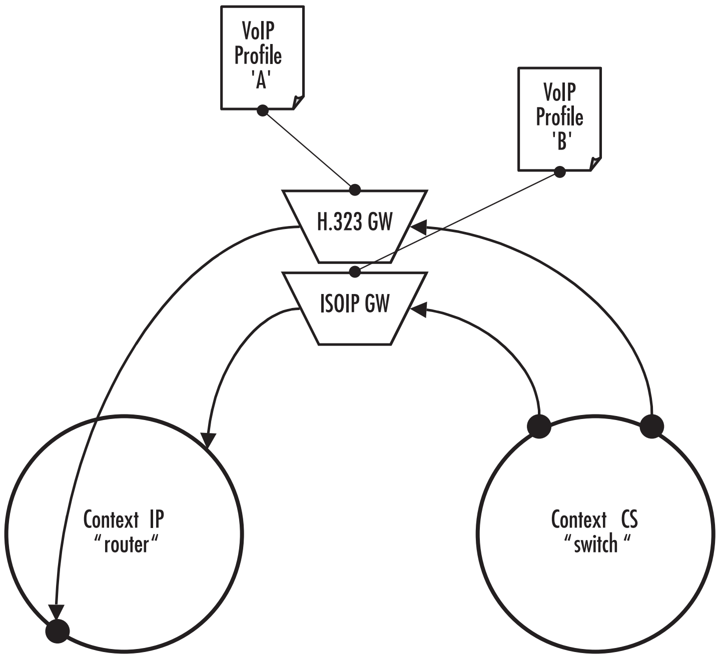


Figure 73. VoIP profile association

As the name implies, configuring voice quality options can improve or degrade the quality of the transmitted voice data. Many of the default values of these components have configured defaults and should only be overwritten 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 various components can be configured in the VoIP profile mode and often in the CS interface mode also. **Generally the configuration in the VoIP profile applies to all calls going through that gateway.** If required, the components could be overwritten for a specific interface. The following configuration tasks describe the configuration of the individual components. You don't have to perform the configuration in the order described below.

- Create a VoIP profile
- Enable DTMF relay (see page 380)
- Enable echo canceller (see page 380)
- Enable silence compression (see page 381)
- Configure voice volume (see page 382)
- Configure dejitter buffer (advanced) (see page 383)
- Enable/disable filters (advanced) (see page 386)
- Configure Fax relaying (see page 386)
- Show VoIP profile configuration and assign to VoIP gateway (see page 390)

Create a VoIP profile

Prior to configuring the voice parameters a VoIP profile has to be created. Each VoIP profile has a name. The name can be any arbitrary string of not more than 25 characters. For ease of identification the name of the associated gateway can be a part of the name. By creating the VoIP profile you enter the VoIP profile configuration mode. Once entered you can configure the various components.

This procedure describes how to create a VoIP Profile and enter the VoIP profile configuration mode

- **Mode:** Configure

Step	Command	Purpose
1	***node*(cfg)#profile voip** *name*	Create a VoIP profile with name *name* and enter.VoIP profile configuration mode
2	***node*(pf-voip)[***name***]#...**	Configuration steps as described in the chapters below

Example: Creating a VoIP profile

This example shows how to create a VoIP profile named ISoIPProfile and enter VoIP profile configuration mode.

```
SN>enable
SN#configure
SN(cfg)#profile voip ISoIPProfile
SN(pf-voip)[ISoIPPr~]#...
```

Enable 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 illustrated in Figure 74. H.323 signals the DTMF tones as H.245 User Input Indications (more precisely as Basic Strings) as most H.323 gateways do.

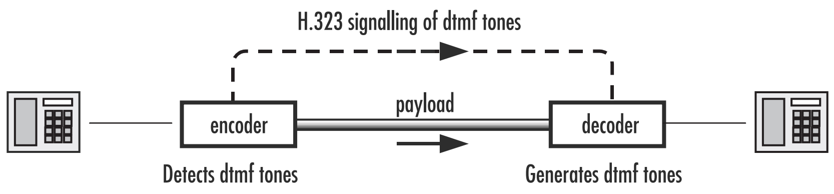


Figure 74. DTMF Relay

This procedure describes how to enable DTMF relay

Mode: Profile VoIP

Step	Command	Purpose
1	***node*(pf-voip)[*name*]#dtmf-relay**	Enable DTMF relay
2	***node*(pf-voip)[*name*]#exit**	Optional: Change to CS interface configuration mode to overwrite the settings in the VoIP profile in a specific CS interface
3	***node*(cfg)#context cs**	
4	***node*(ctx-cs)[switch]#interface isoip** *name* or ***node* (ctx-cs)[switch]#interface h323** *name*	
5	***node*(if-type)[*if-name*]#no dtmf-relay**	Disable DTMF relay on a specific interface

Example: Enabling DTMF relay

The following example shows how to enable DTMF relay for the VoIP profile and disable it for the ISoIP interface.

```
SN>enable
SN#configure
SN(cfg)#profile voip ISoIPProfile
SN(pf-voip)[ISoIPPr~]#dtmf-relay
SN(pf-voip)[ISoIPPr~]#exit
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface isoip IpBackbone
SN(if-isoip)[IpBackb~]#no dtmf-relay
```

Enable echo canceller

Echoes are reflections of the transmitted signal, which result from impedance mismatch in the hybrid (bi-directional 2-wire to 4-wire converting) device. Some voice devices unfortunately have an echo on their wire. Echo cancellation provides near-end echo compensation for this device as illustrated in figure 75.

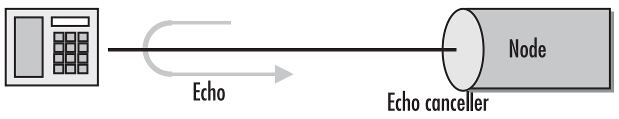


Figure 75. Echo Cancellation

This procedure describes how to enable echo cancellation.

Mode: Profile VoIP

Step	Command	Purpose
1	***node*(pf-voip)[*name*]#echo-canceller**	Enable echo canceller
2	***node*(pf-voip)[*name*]#exit**	Optional:
3	***node*(cfg)#context cs**	Change to CS interface configuration mode to overwrite the settings in the VoIP profile in a specific CS interface
4	***node*(ctx-cs)[switch]#interface isoip** *name* or ***node*(ctx-cs)[switch]#interface h323** *name*	
5	***node*(if-*type*)[*if-name*]#no echo-canceller**	Disable echo canceller on a specific interface

Example: Enable echo canceller

The following example shows how to enable the echo cancellation for the VoIP profile.

```
SN>enable
SN#configure
SN(cfg)#profile voip ISoIPProfile
SN(pf-voip)[ISoIPPr~]#echo-canceller
```

Enable silence compression

Silence compression detects the silent periods in a phone conversation and stops the sending of media packets during these periods. Since conversations are essentially half duplex (people do not talk simultaneously) this can reduce the bandwidth consumption of a voice over packet connection considerably. Comfort noise is generated at the remote end of the silent direction to avoid the impression that the connection is dead. Silence compression and comfort noise can only be enabled together.

This procedure describes how to enable silence compression.

Mode: Profile VoIP

Step	Command	Purpose
1	***node***(**pf-voip**)[*name*]**#silence-compression**	Enable silence compression and comfort noise
2	***node***(**pf-voip**)[*name*]**#exit**	Optional:
3	***node***(**cfg**)**#context cs**	
4	***node***(**ctx-cs**)**[switch]#interface isoip** *name* or ***node***(**ctx-cs**)**[switch]#interface h323** *name*	Change to CS interface configuration mode to overwrite the settings in the VoIP profile in a specific CS interface
5	***node***(**if-***type*)[*if-name*]**#no silence-compression**	Disable silence compression and comfort noise on a specific interface

Example: Enable silence compression

The following example shows how to enable the silence compression for the VoIP profile.

```
SN>enable
SN#configure
SN(cfg)#profile voip ISoIPProfile
SN(pf-voip)[ISoIPPr~]#silence-compression
```

Configure voice volume

The voice volume determines the voice output volume gain towards CS context as illustrated in figure 76.

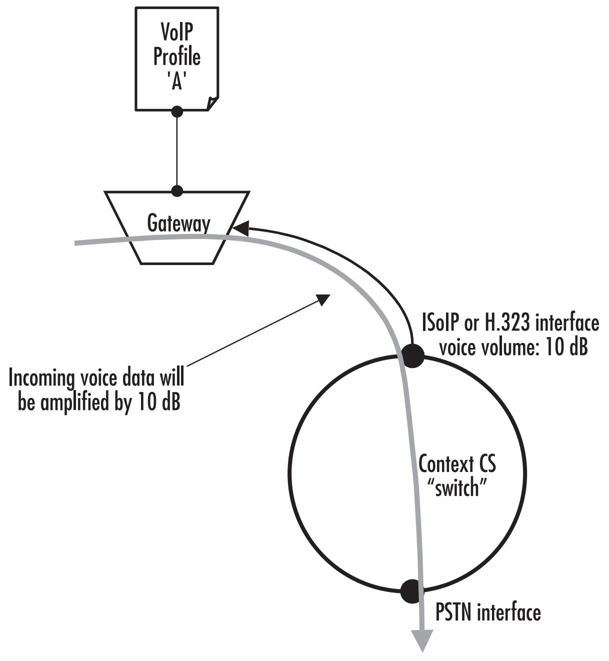

VoIP Profile 'A'
Gateway
ISoIP or H.323 interface voice volume: 10 dB
Incoming voice data will be amplified by 10 dB
Context CS "switch"
PSTN interface

Figure 76. Applying voice volume

This procedure describes how to configure voice volume.

Mode: Profile VoIP

Step	Command	Purpose
1	***node*(pf-voip)[*name*]#voice-volume value**	Set the voice volume to value in dB
2	***node*(pf-voip)[*name*]#exit**	Optional: Change to CS interface configuration mode to overwrite the settings in the VoIP profile in a specific CS interface
3	***node*(cfg)#context cs**	
4	***node*(ctx-cs)[switch]#interface isoip** *name* or ***node*(ctx-cs)[switch]#interface h323** *name*	
5	***node*(*if-type*)[*if-name*]#voice-volume** *value*	Set the voice volume to a different value as specified in the VoIP profile

Example: Configure voice volume

The following example shows how to set the voice volume for the VoIP profile to 10 dB and specify an other value 20 dB for a specific H.323 interface

```
SN>enable
SN#configure
SN(cfg)#profile voip ISoIPProfile
SN(pf-voip)[ISoIPPr~]#voice-volume 10
SN(pf-voip)[ISoIPPr~]#exit
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface h323 IpBackbone
SN(if-h323)[IpBackb~]#voice-volume 20
```

Configure 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 has implemented a dejitter buffer in its voice gateway. 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. The result: packets that are sent in equal spacing from the left gateway arrive with irregular spacing at the right hand gateway, as shown in figure 77

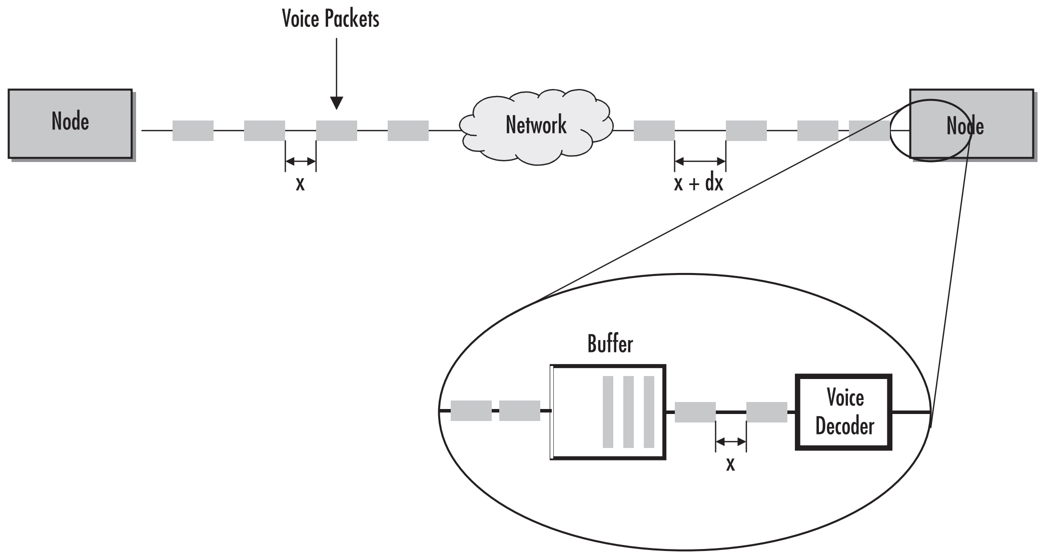


Figure 77. Jitter and dejitter buffer

Since the receiving decompression algorithm (voice decoder) requires fixed spacing between the packets, the typical solution is to implement a dejitter buffer within the gateway. The dejitter buffer deliberately delays incoming packets in order to present them to the decompression algorithm at fixed spacing. It will also fix any out-of-order errors by looking at the sequence number in the RTP frames.

Such a buffer has the effect of smoothing the packet flow, increasing the resiliency of the codec to packet loss, delayed packets and other transmission effects. However, the downside of the dejitter buffer is that it can add significant delay. The dejitter buffer size is configurable and can be optimized for given network conditions. The dejitter buffer size is usually set to be an integral multiple of the expected packet inter-arrival time in order to buffer an integral number of packets. It is not uncommon to see dejitter buffer settings approaching 80 ms for each direction.

SmartWare offers two operation modes for the dejitter buffer, adaptive and fixed, as illustrated in figure 78.

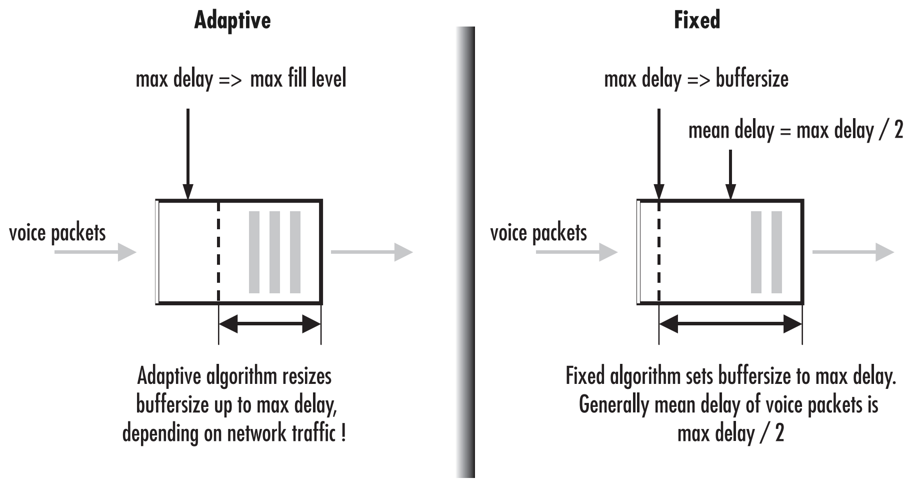

Adaptive
max delay => max fill level
voice packets
Adaptive algorithm resizes buffersize up to max delay, depending on network traffic !
Fixed
max delay => buffersize
mean delay = max delay / 2
voice packets
Fixed algorithm sets buffersize to max delay. Generally mean delay of voice packets is max delay / 2

Figure 78. Adaptive versus fixed dejitter buffer

The adaptive buffer automatically adapts to the network delay variation characteristics and in general yields the best results. The manual fixed buffer is useful only if you have specific information about your network, such as jitter period, etc. and should only used by experienced users.

Note In the adaptive dejitter buffer several more parameters could be configured, such as shrink-speed or grow-step, etc. The default values for these options should only be overwritten if essential. Misconfiguration may lead to interoperability problems and loss of service. Therefore it is strongly recommended that only experienced users change this parameters

This procedure describes how to configure dejitter buffer.

Mode: Profile VoIP

Step	Command	Purpose
1	***node***(**pf-voip)**[*name*]**#dejitter-mode** *mode*	Specify the dejitter buffer as adaptive or fixed.
2	***node***(**pf-voip)**[*name*]**#dejitter-max-delay** *max-delay*	Specify max delay for adaptive or fixed buffer in ms.
3	***node***(**pf-voip)**[*name*]**#exit**	Optional: Change to CS interface configuration mode to overwrite the settings in the VoIP profile in a specific CS interface
4	***node***(**cfg)#context cs**	
5	***node***(**ctx-cs)[switch]#interface isoip** *name* or ***node***(**ctx-cs)[switch]#interface h323** *name*	
6	***node***(*if-type*)[*if-name*]**#dejitter-mode** *mode*	Specify the dejitter buffer for a specific H.323 or ISoIP interface
7	***node***(*if-type*)[*if-name*]**#dejitter-max-delay** *max-delay*	Specify max delay for the dejitter buffer for the specific H.323 or ISoIP interface

Example: Configure dejitter buffer

The following example shows how to define an adaptive dejitter buffer with a maximum delay of 80 ms for the VoIP profile and 40 ms for a specify ISoIP interface

```
SN>enable
SN#configure
SN(cfg)#profile voip ISoIPProfile
SN(pf-voip)[ISoIPPr~]#dejitter-mode adaptive
SN(pf-voip)[ISoIPPr~]#dejitter-max-delay 80
SN(pf-voip)[ISoIPPr~]#exit
SN(cfg)#context cs
SN(ctx-cs)[switch]#interface isoip AccessGateway
SN(if-isoip)[AccessG~]#dejitter-mode adaptive
SN(if-isoip)[AccessG~]#dejitter-max-delay 40
SN(if-isoip)[AccessG~]#
```

Enable/disable filters (advanced)

The voice decoder output is normally filtered using 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 includes several tandems of SmartNodes as illustrated in figure 79, sequential post filtering or high pass filtering can cause quality degradation. In this case, the user can choose to disable the post- and the high-pass-filter.

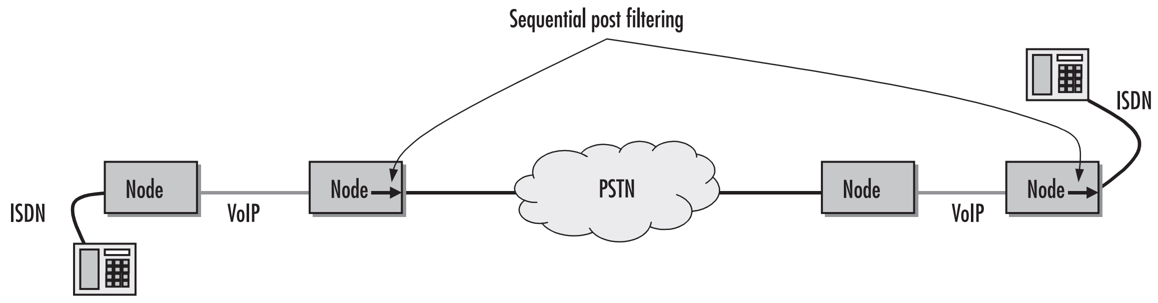


Figure 79. Multiple tandem and sequential post filtering

Note Filtering occurs only with G.723 and G.729 codecs and could only be configured for an VoIP profile and not for an H.323 or ISoIP interface.

This procedure describes how to disable post and high-pass filtering.

Mode: Profile VoIP

Step	Command	Purpose
1	***node***(**pf-voip)**[*name*]**#no post-filter**	Disable decoder output filter
2	***node***(**pf-voip)**[*name*]**#no high-pass-filter**	Disable decoder input high pass filter

Example: Disable filters

The following example shows how to disable the decoder output post-filter and the input high-pass filter of the VoIP profile.

```
SN>enable
SN#configure
SN(cfg)#profile voip ISoIPProfile
SN(pf-voip)[ISoIPPr~]#no post-filter
SN(pf-voip)[ISoIPPr~]#no high-pass-filter
```

Configure Fax and modem handling

Fax is a protocol for electronically transmitting written material in-band over a voice channel. In traditional public switched telephone networks (PSTN) it is handled as well as pure voice. Traditional fax devices scan written material and transform it in tones which are transmitted over the telephone network. The receiver puts together the tones and constructs back the written materials. In IP networks several problems appear which impede to handle fax-data similarly to voice streams:

- If one or more RTP packets which transports 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 data. Compress and decompressing always incur loss of data.
- Dejitter buffers are optimized to force little voice delay, therefore data could be lost.

SmartWare implements two solutions to solve this problems on fax transmission:

- Fax (or modem) bypass—Selects automatically a fallback codec with no or little compression and reconfigures the dejitter buffer for sending and receiveing fax.
- 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- and error-rate than bypass, but the remote endpoint must support the same fax protocol as well.

Figure 80 illustrates the difference between Fax relay and Fax bypass.

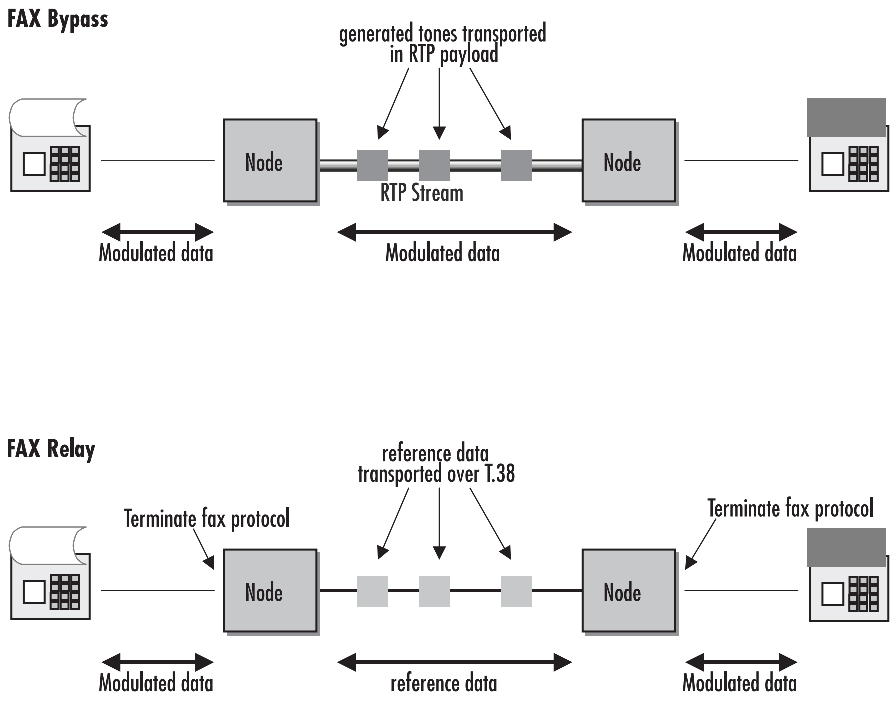


Figure 80. Fax relay and Fax bypass

The commands described below offer more functionality than fax only. Modem connections could be configured in the same manner as fax. For a list of all available protocols use the CLI help.

This procedure describes how to configure fax bypass.

Mode: Profile VoIP

Step	Command	Purpose
1	***node***(**pf-voip)**[*name*]**#{ fax | modem } transmission bypass { g711alaw64k | g711ulaw64k | g726_32k | g726_40k }**	Enable bypass mode and select a codec to be used for fax or modem transmission.
2 optional	***node***(**pf-voip)**[*name*]**#{ fax | modem } dejitter-max-delay** *buffer-size*	For proper operation a dejitter buffer is used on the receiver. The dejitter period can be adjusted to compensate for the jitter imposed by the network. The default value is 200ms which should be ok for almost any transmission network. Only in exceptional cases it may be necessary to increase this value (maximum 400ms). The dejitter buffer implicitly applies the operation mode 'static-data', i.e. minimizes the packet loss.

Example: Enable fax bypass

The following example shows how to configure fax bypass with codec G.711.

```
SN>enable
SN#configure
SN(cfg)#profile voip ISoIPProfile
SN(pf-voip)[ISoIPPr~]#fax transmission bypass g726_32k
```

This procedure describes how to configure fax relay as per T.38

Mode: Profile VoIP

Step	Command	Purpose
1	***node***(**pf-voip)**[*name*]**# fax transmission relay t38-udp**	Enable relay mode with T.38 fax relay protocol using UDP. Relay mode is not available for modem protocols.
2 (optional)	***node***(**pf-voip)[*name*]#fax redundancy ls** *low-speed-redundancy* **hs** *high-speed-redundancy*	Packet loss can be catered for by means of transmitting the fax data packets several times. This can be configured for both the low-speed and the high-speed traffic separately. The default for both parameters is 0 (no redundant transmission). Note that values >0 provide more reliable transmission, but consume additional bandwidth.

Step	Command	Purpose
3 (optional)	***node*(pf-voip)[*name*]# fax dejitter-max-delay** *buffer-size*	For proper operation a dejitter buffer is used on the receiver. The dejitter period can be adjusted to compensate for the jitter imposed by the network. The default value is 200ms which should be ok for almost any transmission network. Only in exceptional cases it may be necessary to increase this value (maximum 400ms). The dejitter buffer implicitly applies the operation mode 'static-data', i.e. minimizes the packet loss.
Step 4 (optional)	***node*(pf-voip)[*name*]#fax volume** *volume*	Adjust 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 how fax transmissions are detected, by the CED tone or by arriving T.38 fax frames (Default: ced-tone).
7 (optional)	***node*(pf-voip)[*name*]#no fax error-correction**	Disables error correction mode (Def: 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 (Def: 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.

Example: Enable Fax relay

The following example shows how to configure fax relay. The max bit rate is increased to 14400 kbps.

```
SN>enable
SN#configure
SN(cfg)#profile voip ISoIPProfile
SN(pf-voip)[ISoIPPr~]#fax transmission relay t38-udp
SN(pf-voip)[ISoIPPr~]#fax redundancy ls 1 hs 2
SN(pf-voip)[ISoIPPr~]#fax dejitter-max-delay 300
SN(pf-voip)[ISoIPPr~]#fax max-bit-rate 9600
SN(pf-voip)[ISoIPPr~]#fax detection fax-frames
```

Show VoIP profile configuration and assign it to a VoIP gateway

To activate a created VoIP profile it must be used by an H.323 or ISoIP gateway. Prior to using the profile by a gateway it is possible to display the VoIP profiles.

Note The default VoIP profile always exists. Settings such as fax/data can be ignored at this time.

This procedure describes how to show the actual VoIP profiles and use it by a VoIP gateway.

Mode: Profile VoIP

Step	Command	Purpose
1	***node***(**pf-voip)[***name***]#show profile voip [name]**	Show all defined VoIP profiles or the VoIP profile with name *name*
2	***node***(**pf-voip)[***name***]#exit**	Change to gateway configuration mode.
3	***node***(**cfg)#gateway** *gw-type gw-name*	
4	***node***(*gw-type*)[*gw-name*]**#use voip-profile** *name*	The VoIP profile with name *name* is used by the VoIP gateway.

Example: Show VoIP profile configuration and assign it to a VoIP gateway

This example shows how to show all defined VoIP profiles and how to use the VoIP profile ISoIP profile by the ISoIP gateway.

```
SN(pf-voip)[ISoIPPr~]#show profile voip

Profile default
-----------------------------------------------------------------
dejitter settings: mode:                        adaptive
                   max delay:                   60 ms
                   max packet loss:             4
                   shrink speed:                1
                   grow step:                   1
                   grow attenuation:            1
voice settings:    hybrid loss:                 6
                   non linear processor mode: adaptive
                   echo canceller:              enabled
                   silence compression:         disabled
```

```
                     high pass filter:          enabled
                     post filter:               enabled
                     volume:                    0 dBm
fax settings:        transmission:              none
                     relay protocol:            T38UDP
                     high-speed t.38 redundancy 0
                     low-speed t.38 redundancy  0
                     bypass codec:              g711alaw64k
                     max bit rate:              14400 bit/s
                     volume:                    -9.5 dBm
                     error correction:          enabled
                     hdlc image transfer:       enabled
                     dejitter max delay         200 ms
modem settings:      transmission:              none
                     bypass codec:              g711alaw64k
                     dejitter max delay         200 ms
DTMF settings:       DTMF relay:                enabled
                     mute encoder:              enabled


Profile ISoIPProfile
------------------------------------------------------------------
dejitter settings: mode:                        adaptive
                     max delay:                 60 ms
                     max packet loss:           4
                     shrink speed:              1
                     grow step:                 1
                     grow attenuation:          1
voice settings:      hybrid loss:               6
                     non linear processor mode: adaptive
                     echo canceller:            enabled
                     silence compression:       disabled
                     high pass filter:          enabled
                     post filter:               enabled
                     volume:                    0 dBm
fax settings:        transmission:              relay
                     relay protocol:            T38UDP
                     high-speed t.38 redundancy 2
                     low-speed t.38 redundancy  1
                     bypass codec:              g711alaw64k
                     max bit rate:              9600 bit/s
                     volume:                    -9.5 dBm
                     error correction:          enabled
                     hdlc image transfer:       enabled
                     dejitter max delay         300 ms
modem settings:      transmission:              none
                     bypass codec:              g711alaw64k
                     dejitter max delay         200 ms
DTMF settings:       DTMF relay:                enabled
                     mute encoder:              enabled

SN(pf-voip)[ISoIPPr~]#exit
SN(cfg)#gateway h323 h323
SN(gw-isoip)[isoip]#use voip-profile ISoIPProfile
SN(gw-isoip)[isoip]#
```

Example

Home office in an enterprise network

Figure 81 is an example of a home office in an enterprise network. The connection bandwidth amounts to 128 kbps and is very low quality. Therefore low-bit-rate codec G.723_5k3 is used which is only supported on ISoIP. Likewise silence compression is also enabled. Because of the low-bit-rate codec dtmf relay is enabled. 80 to 100 ms jitter is expected, therefore the dejitter buffer is set to adaptive with a maximum delay of 100 ms. Because the wire causes a high alternation of the voice signal, voice volume gain has to be set to 20 dB.

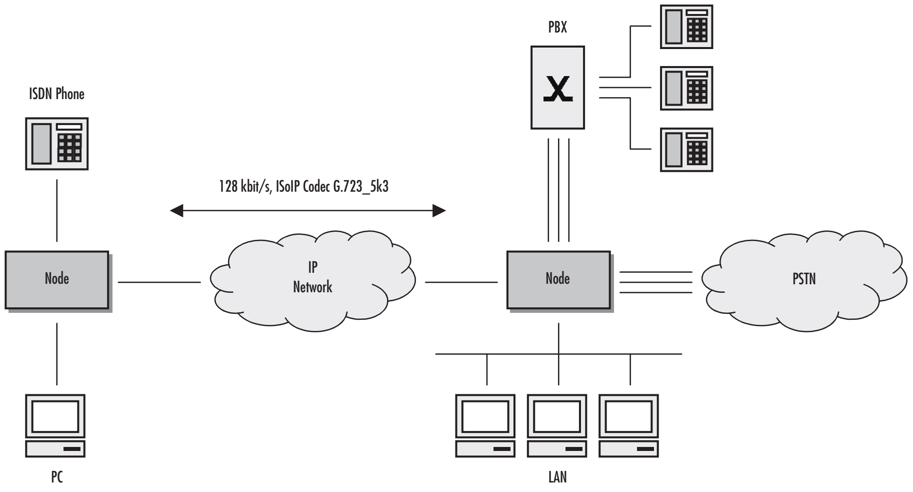


Figure 81. Home office in an enterprise network

First we configure the required CS interfaces and call routing. In this chapter we focus on configuring VoIP profiles, therefore we have omitted the configuration of the CS interfaces and call routing.

Next we configure the voice over IP settings as needed from the description above. First we create the VoIP profile with the needed configurations.

```
SN>enable
SN#configure
SN(cfg)#profile voip Wire128kbit
SN(pf-voip)[Wire128~]#silence-compression
SN(pf-voip)[Wire128~]#dtmf-relay
SN(pf-voip)[Wire128~]#dejitter-mode adaptive
SN(pf-voip)[Wire128~]#dejitter-max-delay 100
SN(pf-voip)[Wire128~]#voice-volume -3
SN(pf-voip)[Wire128~]#exit
SN(cfg)#
```

Afterwards we show the configuration and configure the ISoIP gateway to use the VoIP profile. As aforementioned fax and data settings may be ignored.

```
SN(cfg)#show profile voip Wire128kbit

Profile Wire128kbit
------------------------------------------------------------------
dejitter settings: mode:                        adaptive
                   max delay:                   100 ms
                   max packet loss:             4
                   shrink speed:                1
                   grow step:                   1
                   grow attenuation:            1
voice settings:    hybrid loss:                 6
                   non linear processor mode: adaptive
                   echo canceller:              enabled
                   silence compression:         enabled
                   high pass filter:            enabled
                   post filter:                 enabled
                   volume:                      -3 dBm
fax settings:      transmission:                none
                   relay protocol:              T38UDP
                   high-speed t.38 redundancy 0
                   low-speed t.38 redundancy  0
                   bypass codec:                g711alaw64k
                   max bit rate:                14400 bit/s
                   volume:                      -9.5 dBm
                   error correction:            enabled
                   hdlc image transfer:         enabled
                   dejitter max delay           200 ms
modem settings:    transmission:                none
                   bypass codec:                g711alaw64k
                   dejitter max delay           200 ms
DTMF settings:     DTMF relay:                  enabled
                   mute encoder:                enabled

SN(cfg)#gateway isoip isoip
SN(gw-isoip)[isoip]#use voip-profile Wire128kbit
SN(gw-isoip)[isoip]#
```

We have omitted the configuration of the ISDN ports and show the configuration of the codec and enabling of the gateway.

```
SN(gw-isoip)[isoip]#codec g723_5k3 20
SN(gw-isoip)[isoip]#no shutdown
```

Last we have to enable the CS configuration. For more information refer to 24, “CS interface configuration” on page 295 on page 295.

Chapter 31 VoIP debugging

Chapter contents

Introduction

This chapter helps you to localize a system component that is responsible for faults during operation of a SmartNode device. This chapter provides debugging strategies to help you locating the origin of an error, and describes the necessary debug and show commands. The emphasis is on VoIP debugging. IP debug commands are described in this chapter concerning the appropriate system component.

This chapter includes the following sections:

- Debugging strategy
- Debugging task list (see page 396)

Typically, SmartWare **show** and **debug** commands are used to provide information to verify correct system operation and to troubleshoot problems. This chapter describes general system-wide monitoring and testing tasks, such as displaying system memory and processes, displaying all system hardware, testing IP and circuit switch connectivity, and enabling debugging messages for all IP packets.

For information on **show** and **debug** commands that are specific to a feature, interfaces, ports, or circuits, see the appropriate chapter in this guide. For example, to find out how to display or debug the Session Router, see chapter 25, "Session router configuration" on page 311.

SmartWare supports file logging. Event logs contain warnings and information from system components of SmartWare. Entries in the logs always have the actual system time. Please make sure that your 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 let it automatically set by SNTP or by ISDN. Refer to chapter 8, "Basic system management" on page 91, chapter 19, "SNTP client configuration" on page 223, or chapter 23, "CS context overview" on page 275.

Debugging strategy

Multi-service IP networks build on highly sophisticated systems and protocols that offer a great many possibilities. Unfortunately the possible sources of trouble are almost as many. Therefore it is important to use a very methodical approach when tracking down a problem:

- Work from the bottom to the top of the protocol stack. Always make sure 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.

Debugging task list

Depending on problem that has occurred one or more of the following debugging tasks should be performed. First verify IP and CS connectivity. The next three tasks describe the debugging of ISDN and VoIP protocols.

The session control data task describes debugging of the data between the ISDN and VoIP protocols, and in the voice over IP data task it is explained how to debug tones and codecs. Finally check event logs, which could help you to find out the cause of a problem.

- Verify IP connectivity
- Verify circuit switch connectivity (see page 398)
- Debug ISDN data (advanced) (see page 401)
- Debug H.323 data (advanced) (see page 401)
- Debug ISoIP data (advanced) (see page 401)
- Debug session control data (advanced) (see page 401)
- Debug Voice over IP data (advanced) (see page 402)
- Check event logs (see page 405)

Verify IP connectivity

With the following command you can verify that an IP packet can be sent to and received from the destination host.

This procedure describes how to test connectivity by verifying IP accessibility of hosts

Mode: Administrator execution

Step	Command	Purpose
1	***node*#ping** *ipaddress* [*number-of-packets*] [**timeout** *seconds*]	Verify whether an IP host is reachable or not

Example: Verify IP connectivity

The following example shows how to test the connectivity to a remote host by verifying the accessibility by the ping command. Furthermore it shows the output if host 192.195.23.1 is not, and host 172.16.40.122 is reachable.

```
SN#ping 192.195.23.1 10 timeout 5
Sending 10 ICMP echo requests to 192.195.23.1, 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
```

Verify circuit switch connectivity

The following commands makes possible to establish voice calls between two endpoints without additional voice devices such as voice phones. Additionally the set up and shut down of voice calls can be traced by the debug monitor and established calls can be displayed by the show command.

This procedure describes how to verify circuit switch connectivity by set up and trace voice calls

Mode: Administrator execution

Step	Command	Purpose
1	***node2***(*if-type*)[*if-name*]**#routing dest-interface callapp**	The call from the H.323 or ISoIP interface must be routed to the interface 'callapp' to accept it with SmartWare.
2	***node2***(*if-type*)[*if-name*]**#exit**	
3	***node2*(ctx-cs)[switch]#no shutdown**	Enable the configuration
4	***node2*(ctx-cs)[switch]#exit**	
5	***node2*(cfg)#exit**	
6	***node2*#debug call** [*detail level*]	Enable call debug monitor on SmartNode number 2.
7	***node1*#debug call** [*detail level*]	Enable call debug monitor on SmartNode number 1.
8	***node1*#call** *callkey* **dial** *interface*	Set up the call with the call ID *callkey* and over the interface *interface* on SmartNode number 1.
9	***node2*#call** *callkey* **accept**	Accept the call on SmartNode number 2. You will get the *callkey* from the debug monitor.
10	***node2*#show call sessions** [*detail level*]	Display the actual running call application session(s). If there is one displayed on both endpoints the connection was established.
11	***node1*#show call sessions** [*detail level*]	
12	***node1*#call** *callkey* **drop**	Tear down the call on both endpoints.

Example: Verify circuit switch connectivity

The following example shows how to establish a voice connection between two endpoints. Note that to perform this example you need a fully operative configuration on both endpoints. The following configuration steps only illustrate how to set up and shut down a voice call.

SmartNode 2: Route calls from ISoIP interface *ISOIP_IF* to interface *callapp* and enable debug monitor.

```
SN2>enable
SN2#configure
SN2(cfg)#context cs
SN2(ctx-cs)[switch]#interface isoip ISOIP_IF
SN2(if-isoip)[ISOIP_IF]#routing dest-interface callapp
SN2(if-isoip)[ISOIP_IF]#exit
SN2(ctx-cs)[switch]#no shutdown
SN2(ctx-cs)[switch]#exit
SN2(cfg)#exit
SN2#
SN2#debug call 5
```

SmartNode 1: Enable debug monitor and dial

```
SN1#debug call 5
SN1#
SN1#call 21 dial ISOIP_IF
SN1#14:12:55  CALL  > [0021] SENT
[080005]
SETUP (Generic Q.931)
  [04039090A3]
  Bearer capability : 3.1kHz Audio - CCITT
    circuit mode - 64 kbps - G.711 A-law
  [700181]
  Called party number :
    unknown number - E.164 numbering plan

14:12:55  CALL  > [0021] GOT
[080001]
ALERTING (Generic Q.931)

14:12:55  CALL  > [0021] ALERTING

SN1#
```

SmartNode 2: Accept call

```
SN2#14:12:56  CALL  > [8000] GOT
[080005]
SETUP (Generic Q.931)
  [04039090A3]
  Bearer capability : 3.1kHz Audio - CCITT
    circuit mode - 64 kbps - G.711 A-law
  [700181]
  Called party number :
    unknown number - E.164 numbering plan

14:12:56  CALL  > [8000] SENT
[080001]
ALERTING (Generic Q.931)

14:12:56  CALL  > [8000] ALERTING

SN2#

SN2#call 8000 accept
SN2#14:13:31  CALL  > [8000] SENT
[080007]
CONNECT (Generic Q.931)

14:13:31  CALL  > [8000] GOT
[08000F]
CONNECT ACKNOWLEDGEMENT (Generic Q.931)

SN2#
```

SmartNode 1: See debug monitor output and show call sessions

```
14:13:30  CALL  > [0021] GOT
[080007]
CONNECT (Generic Q.931)

14:13:30  CALL  > [0021] SENT
[08000F]
CONNECT ACKNOWLEDGEMENT (Generic Q.931)

14:13:30  CALL  > [0021] CONNECTED

SN1#
SN1#show call sessions 5
[0021]
SN1#


SmartNode 2: Show call sessions

SN2#show call sessions 5
[8000]
SN2#
```

SmartNode 1: Drop call and see debug monitor output

```
SN1#call 21 drop
SN1#14:14:49  CALL  > [0021] SENT
[080045]
DISCONNECT (Generic Q.931)
  [0803038090]
  Cause : normal call clearing
    transit network - CCITT - Q.931

14:14:49  CALL  > [0021] SENT
[08004D]
RELEASE (Generic Q.931)
  [0803038090]
  Cause : normal call clearing
    transit network - CCITT - Q.931

14:14:49  CALL  > [0021] DROPPED

SN1#
```

SmartNode 2: See debug monitor output

```
SN2#14:14:50  CALL  > [8000] GOT
[080045]
DISCONNECT (Generic Q.931)
  [0803038090]
  Cause : normal call clearing
    transit network - CCITT - Q.931

14:14:50  CALL  > [8000] SENT
```

```
[08004D]
RELEASE (Generic Q.931)
  [0803038090]
  Cause : normal call clearing
    transit network - CCITT - Q.931

14:14:50  CALL  > [8000] DROPPED

SN2#
```

Debug ISDN data

To obtain actual information about the ISDN layers you can use the ISDN debug commands. The show command displays information about the actual ISDN status and about actual ISDN sessions. The debug command enables the ISDN debug monitor to show layer 1 to layer 3 information. This information is mainly useful for experienced users.

This procedure describes how to get information about ISDN data

Mode: Administrator execution

Step	Command	Purpose
1	***node*#show isdn bearer-channels** [*detail-level*]	Show the actual ISDN status
2	***node*#show isdn sessions** [*detail-level*]	Show the actual status of ISDN sessions
3	***node*#debug isdn** [*slot*] [*port*] [*layer*]	Enable the ISDN debug monitor to get layer 1 to layer 3 information.

Debug H.323 data

You can use the H.323 debug commands if there are problems with setting up and establishing calls over H.323. The show command displays the actual H.323 configuration or the actual status of the H.323 stack, e.g. ongoing calls or registration state. The debug command enables the H.323 debug monitor to show specific information for a call, such as status, used codecs or transposed messages. There are more than forty different debug monitors such as 'q931', 'ras' or 'signaling'. For a list of all available debug monitors refer to the CLI online help.

This procedure describes how to get information about H.323 data

Mode: Administrator execution

Step	Command	Purpose
1	***node*#show gateway h323 config**	Show the actual H.323 configuration
2	***node*#show gateway h323 status**	Show the actual status of the H.323 stack.
3	***node*#debug gateway h323 signaling** [*detail-level*]	Enable the H.323 debug monitor to get information about the transposed H.323 data. Debug monitor 'signaling' is mostly used.

Debug session control data

Voice application data contains voice data that is transposed between ISDN and VoIP protocols. The show command displays registered subsystems (e.g. ISoIP and Datapath) or open VoIP data sessions. The debug command enables the session-control debug monitor to show transposed data between ISDN and VoIP protocols. Additionally a VoIP session could be closed manually by a command.

This procedure describes how to obtain information about session-control.

Mode: Administrator execution

Step	Command	Purpose
1	***node#*show session-control subsystems** [*detail-level*]	Show registered subsystems
2	**node#show session-control sessions** [*detail-level*]	Show open VoIP data sessions
3	***node#*debug session-control** [*detail-level*]	Enable the session-control debug monitor to show exchanged data between ISDN and VoIP protocols.
4	***node#*session-control close** [*session-id*]	Close an established VoIP session manually, enter 'all' to close all sessions.

Debug ISoIP data

You can use the ISoIP debug commands if there are problems with setting up and establishing calls over ISoIP. The show command displays information about ISoIP connections and sessions. An ISoIP connection contains one ore more ISoIP sessions. Generally an ISoIP connection is set up and is then used for several ISoIP sessions. If no session establishes for about two minutes the ISoIP connection is closed.

The debug command enables the H.323 debug monitor to show specific information for establishing and closing ISoIP connections.

This procedure describes how to obtain information about ISoIP data

Mode: Administrator execution

Step	Command	Purpose
1	***node#*show gateway isoip connections** [*detail-level*]	Show the actual status of ISoIP connections.
2	***node#*show gateway isoip sessions** [*detail-level*]	Show the actual status of ISoIP sessions
3	***node#*debug gateway isoip** [*detail-level*]	Enable the ISoIP debug monitor to get information about ISoIP connections.

Debug Voice over IP 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 establishing of voice connection, no or wrong tone 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 (underrun, 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 the SmartNode): 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 Protocl) 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	***node*#debug voip events**	**Enable the voip events monitor .**
2	***node*#debug dsp error**	**Enable the DSP error monitor.**
3	***node*#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
***node*#debug dsp events**	**Enable the DSP events monitor .**
***node*#debug voip dejitter**	**Enable the dejitter buffer monitor.**
***node*#debug voip rtp**	**Enable the RTP debug monitor**
***node*#debug voip cs**	**Enable the CS debug monitor**

Example: Debugging Fax connections

Symptoms: Fax transmission starts but is interrupted and the Fax machines terminate wit an error message, Fax transmission does not start at all, SmartNode 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 sections "Debug H.323 data" on page 401 and "Debug session control data" on page 401).

Attention: Special signaling procedures are used for the transition between voice and fax data transmission. Possibly the initial call setup is correct, but the signaling to T.38 over H.323 is bogus. 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	***node*#debug voip events**	Enable the voip events monitor .
2	***node*#debug voip t38 events**	Enable the T.38 events monitor
3	***node*#debug voip t38 dejitter**	Enable the T.38 dejitter monitor
4	***node*#debug dsp error**	Enable the DSP error monitor.

Step	Command	Purpose
5	***node*#debug dsp t30**	Enable the T.30 flow monitor

Depending on the type of problem, use also:

Command	Purpose
***node*#debug dsp events**	Enable the DSP event monitor
***node*#debug dsp ifp**	Enable the IFP decoding monitor

Check system logs

See section "Displaying the system logs" on page 97.

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). When added to the submitted trouble report this 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 the SmartNode(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'. That the logs are useful, it is necessary to set on startup 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 SmartWare and can be started (in a Telnet and/or console session) individually ('debug' command).

  Note In order to correlate the protocol monitors at the different levels in SmartWare (e.g. ISDN layer3 and Session-Router monitors) run the monitors concurrently.

- **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: **international.sales@inalp.ch** (use fax only in exceptional cases).

Appendix A **Terms and definitions**

Chapter contents

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 relate 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 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 *release*.
Call Routing	Calls through SmartNode can be routed based on a set of routing criteria. See also Ses*sion Rou*ter.
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 *Silence Compression*.
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 SmartWare CLI commands, which are used to configure the software modules of 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, SmartWare includes a dejitter buffer. Storing packets in a dejitter buffer before they are transferred to the local ISDN equipment, e.g. telephone, SmartWare converts a variable delay into a fixed delay, giving voice a better quality. See also *Jitter*.
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. 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 Canceller	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 *flash memory.*
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 ISoIP 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 *post filter*
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 *node*.
Hostname	Name given to a computer system, e.g. a PC or workstation.
Hunt Group	In the 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 *Q.SIG*
ISoIP	ISDN over IP is patent pending solution of Patton Electronics to carry ISDN services over IP networks.
Jitter	Jitter is the variation on packets arriving on a SmartNode. See also *dejitter buffer.*
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 *High-Pass Filter*.
POTS	Plain Old Telephone Service
Profile	A profile provides configuration shortcutting. A profile contains specific settings which 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 *ISDN Services*.
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 *build*.
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	The SmartNode is Patton Electronics' networking product available in three series: • 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. 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.
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.
System Memory	The volatile memory, that includes the system: region, holding the running-config for SmartWare during operation of a SmartNode.

Term or Definition	Meaning
system:	A region in the volatile memory of a SmartNode. See also *system memory.*
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 Configuration mode overview

Chapter contents

Introduction

figure 82 on page 418 illustrates the configuration modes hierarchy. Each box contains the mode name, the enter command and the prompt in a telnet console. Additionally all relationships between the instances of the components through bind and link commands are illustrated. For example an instance of 'port ethernet' must be bound to an 'IP interface' through the command '[no] bind interface <name> [<ip_context>]'.

SmartWare 2.20 command summary

The SmartWare 2.20 commands are collected in configuration modes as illustrated figure 82. Following all commands in the configuration modes are listed. The configuration modes are listed in order as shown in figure 82. 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 (|).

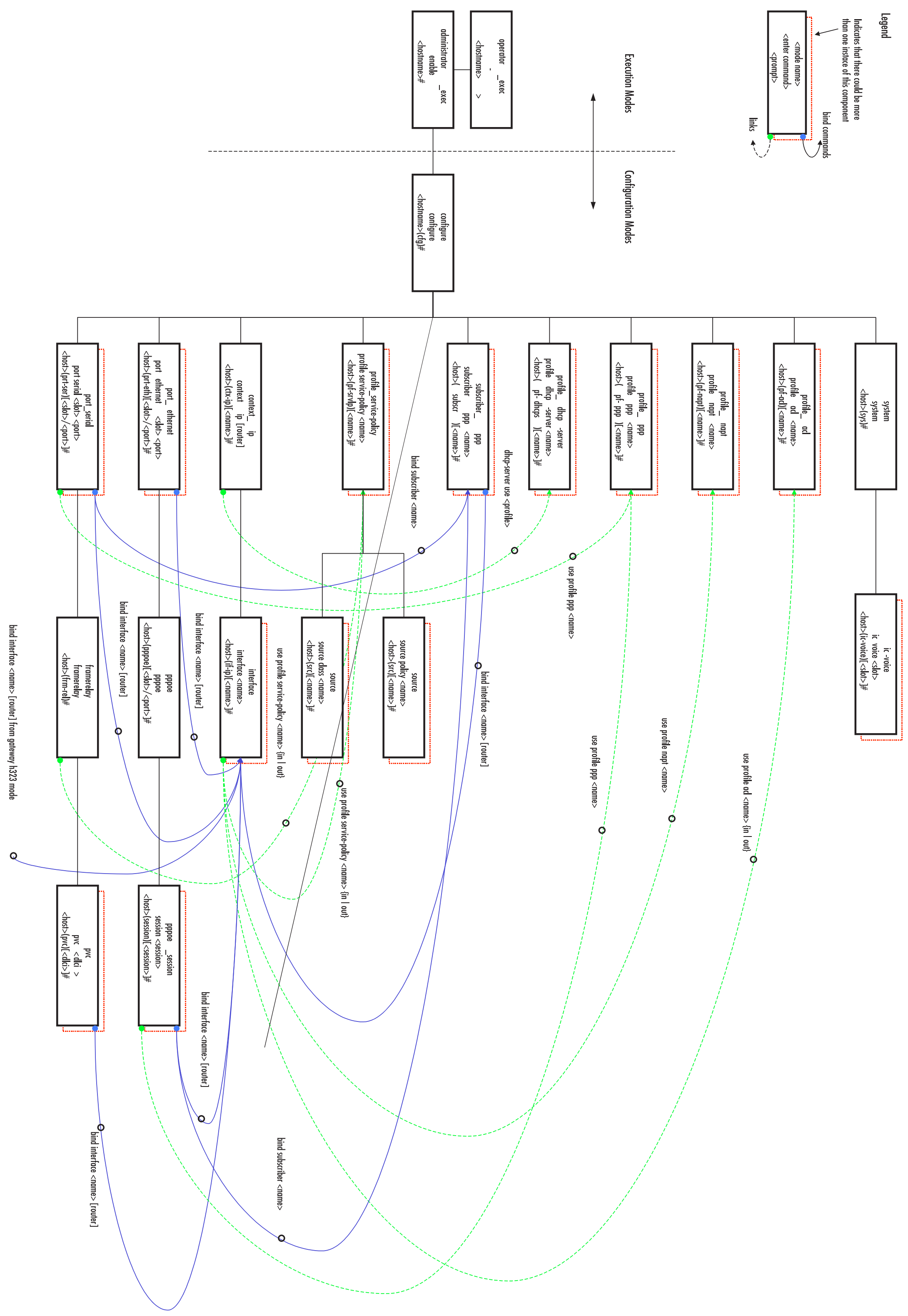


Figure 82. Configuration modes and bind and link commands overview

Command syntax is illustrated by an example in figure 83.

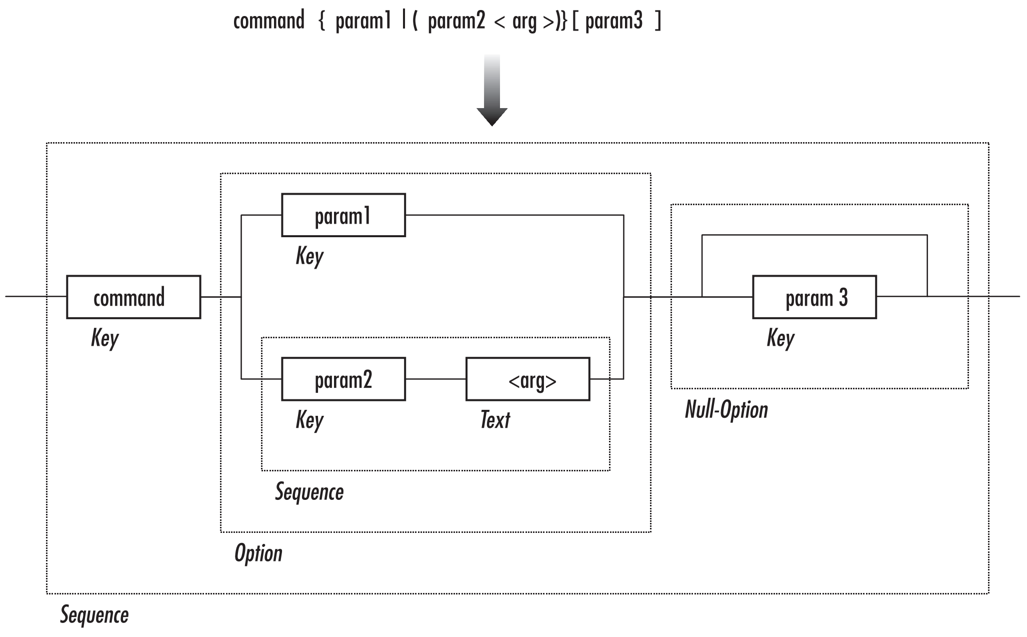


Figure 83. EBNF syntax

This command summary is valid for *SmartWare, Release 2.20, Build 22126*. Commands in future SmartWare releases or builds may be different. The information provided in this chapter is subject to change without notice.

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 {(<callkey> {(dial <interface> [<called-party> [<calling-party> ] ] ) |
     (overlap <called-party> ) | accept | drop | (display <display-data> ) | (keypad
     <keypad-data> ) | (user <user-data> ) | hold | (suspend [<parkcode> ] ) |
     retrieve | (resume [<parkcode> ] ) } ) | autoaccept | (bearer-capability {audio
     | speech | digital } ) | ({called-numbering-plan | calling-numbering-plan }
     {e164 | private } ) | ({called-type-of-number | calling-type-of-number }
     {unknown | national | international | subscriber } ) }
clear
show call {sessions | config } [<detail> ]
[no] debug call [<detail> ]
show port {pstn | isdn } [<detail> ]
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 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 ethernet [<print-slot> <print-port> ]
show port serial [<print-slot> <print-port> ]
show framerelay [pvc <print-dlci> ]
show ppp {links|networks} [<level> ]
show pppoe [<level> ]
show log [{event|reset|boot} ]
show service-policy [interface <interface-name> [router ] ]
show version
{(jobs ) | (fg <job> ) }
terminal width {<value> | default }
terminal height {<value> | default }
show terminal
[no] terminal more
[no] terminal idle-time-logout <value>
logout
su <account>
who
help
show history
show version cli
show smi
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> | <src> } {{running-config|startup-config|sys-
     tem:running-config|flash:} | {cli:|preferences:} | <dest> | <dest> }
erase {{startup-config} | {cli:|preferences:} | <config> }
edit <file>
debug all
[no] debug gateway isoip [isoip ] [<detail> ]
show gateway isoip [isoip ] {sessions | connections } [<detail> ]
isoip {(connect <destip> ) | (disconnect <destip> ) }
[no] debug {pstn | isdn } <slot> <port> {all | layer1 | layer2 | layer3 }
[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 voip rtp
[no] debug voip demux
[no] debug voip cs
[no] debug voip t38 events
[no] debug voip t38 dejitter
[no] debug snmp private-mib
[no] debug h235security [<detail>]
```

```
[no] debug gateway h323 [h323 ] {{signaling | ras | h245 } | (stack {appl | cat | cm
     cmapi cmapicb cmerr debug efrm li liinfo namec han pdalapi pdlcomm pdlconf
     pdlncode pdlerror pdlfnerr pdlprint pdlprnerr pdlsm pdlmisc pdlmtask pdllist
     pdltimer per pererr q931 ra rasc trl rasindb seli timer tpktchan tunnctrl
     udpchan unreg vt] [<detail>]
[no] debug session-control [switch ] [<detail> ]
dsp {(up [<slot> ] ) | (down [<slot> ] ) | (reconfigure [<slot> ] ) | (test [<slot>
     ] ) | (channel {(restart [<slot> [<channel> ] ] ) | (state {(reset [<slot>
     [<channel> ] ] ) | (closed [<slot> [<channel> ] ] ) | (opened [<slot> [<chan-
     nel> ] ] ) } ) } ) | (tone {(play <slot> <channel> {<name> | <id> } ) | (stop
     <slot> <channel> ) } ) | (dtmf {(play <slot> <channel> <digit> ) } ) }
[no] debug cli
[no] debug debug
[no] debug acl [{in | out} [<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-control|tx-
     option|timer|state-machine|controlmanagement|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|tx-error|lmi|packet|management} ]
[no] debug ipsec
[no] debug db
session-control close <session>
show accounts
show {nvram: | {running-config|factory-config|startup-config|system:running-con-
     fig} | {cli:|preferences:} | <config> }
show crc {{running-config|factory-config|startup-config|system:running-config} |
     {cli:|preferences:} | <config> }
show subscriber ppp [<subscriber_ppp_name_show> ]
show profile ppp [<profile_ppp_name_show> ]
show ipsec security-associations
show ipsec policy manual
show profile ipsec-transform
show profile service-policy[<arbiter-name>]
show (h235security)
show {(gateway h323 [h323 ] {config | status | stack-config } ) }
show {pstn | isdn } {sessions | layer3-status | bearer-channels } [<detail> ]
show session-control [switch ] {subsystems | sessions } [<detail> ]
show context cs [switch ] {config | orphans | monkeys } [<detail> ]
show snmp
show sntp-client
show profile acl [<acl_name> ]
reload
show pluginframe repository
{(show flashserver {general | sector | file-part-list | file-list | error-file-list
     } ) | (flashserver {garbage_removing } ) }
```

```
crash {alignment|illegal-opcode|privileged-instr|trap|low-memory|over-
     load|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> ) }
spr {(get <register> ) | (set <register> <value> ) }
show task
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>
[no] administrator <account> password <password>
[no] operator <account> password <password>
[no] banner <banner>
clock set <time>
[no] webserver [port <port> ] [lang {en|de} ]
[no] snmp community <community> {ro|rw}
[no] snmp target <ipAddress> security-name <community>
[no] snmp host <ipAddress> security-name <community>
[no] sntp-client
sntp-client server {primary|secondary} <server_address> [port <sntp_port> ] [version
     <version_number> ]
sntp-client operating-mode {unicast | multicast | anycast }
sntp-client anycast-address <ip_anycast-address> [port <sntp_port> ]
sntp-client local-port <sntp_port>
sntp-client poll-interval <number_pollinterval>
[no] sntp-client local-clock-offset
[no] sntp-client root-delay-compensation
sntp-client gmt-offset {+|-} <time_gmtoffset>
[no] sntp-client secure-mode
[no] sntp-client authentication
system hostname <string>
system location <string>
system contact <string>
system supplier <string>
system provider <string>
system subscriber <string>
```

system

```
system
```

```
[no] bypass-mode
clock-source {internal | (<slot> <port> ) }
[no] synchronize-to-isdn-time
[no] local-inband-tones
```

ic_voice

```
ic voice <slot>
pcm {(law-select {aLaw | uLaw } ) }
exit
exit
```

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|con-
     version-error|dod-host-prohibited|dod-net-prohibited|echo|echo-reply|general-
     parameter-problem|host-isolated|host-precedence-unreachable|host-redi-
     rect|host-tos-redirect|host-tos-unreachable|host-unknown|host-unreach-
     able|information-reply|information-request|mask-reply|mask-request|mobile-
     redirect|net-redirect|net-tos-redirect|net-tos-unreachable|net-unreach-
     able|network-unknown|no-room-for-option|option-missing|packet-too-big|parame-
     ter-problem|port-unreachable|precedence-unreachable|protocol-
     unreachable|reassembly-timeout|redirect|router-advertisement|router-solicita-
     tion|source-quench|source-route-failed|time-exceeded|timestamp-reply|times-
     tamp-request|traceroute|ttl-exceeded|unreachable} type <type> type <type> code
     <code> ] ) } [[tos {max-reliability|max-throughput|min-delay|min-monetary-
     cost|normal} ] [precedence {{critical|flash|flash-override|immediate|inter-
     net|network|priority|routine} | <precedence> } ] dscp <dscp> [mask
     {1|3|7|15|31} ] ] [cos <cos> cos-rtp <cos_rtp> <cos_rtcp> ] [ipsec-pol-
     icy<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> ]
[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 [<value> ]
```

source

```
[no] source {(class <source-name> ) | (policy <source-name> ) }
rate {<value> | remaining }
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 [<value> ]
    exit
    exit
```

profile_napt

```
    [no] profile napt <napt-profile_name>
    [no] icmp default <host>
    [no] range <local_ipl> <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-nak request interval <interval> max <max>
    lcp-echo-request interval <interval> max <max>
    mtu min <min> max <max>
    mru min <min> max <max>
    accm <min> max <maxvalue>
    [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>
    exit
```

ipsec manual policy

```
    [no] profile ipsec-policy-manual
    [no] use profile ipsec-transform
    [no] session-key {inbound|outbound} {ah-authentication|esp-authenticatoin|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 <internal_tone_name> <call_progress_tone_name> ) }
exit
```

profile_voip

```
[no] profile voip <name>
hybrid-loss {0|3|6|9}
non-linear-processor-mode {adaptive|disabled|silence}
[no] high-pass-filter
[no] post-filter
[no] dtmf-relay
[no] dtmf-mute-encoder
[no] echo-canceller
[no] silence-compression
[no] voice-volume <voice_volume>
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 {(relay t38-udp ) | (bypass {g711alaw64k|g711ulaw64k} ) }
fax redundancy ls <ls_red> hs <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 (bypass {g711alaw64k|g711ulaw64k} )
modem dejitter-max-delay <buffer_size>
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
```

context_ip

```
    context ip [router ]
    [no] dhcp-server
    dhcp-server clear-lease {all | <address> }
    [no] dhcp-server use <name>
    [no] route <destaddr> <destmask> {<gwaddr> | <interface> } [<metric> ]
    [no] icmp rtp-port-range
```

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 <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
    exit
```

subscriber_ppp

```
    [no] subscriber ppp <subscriber_ppp_name>
    dial {in|out}
    [no] authentication {(chap pap ) | {chap|pap} }
    [no] identification {outbound|inbound} <id> [password <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
```

context_cs

```
context cs [switch ]
delete {all | all-functions | all-routing-tables | all-translation-tables | all-
     interfaces }
[no] number-prefix {national | international } <prefix>
[no] called-party <name> <key> ({({(dest-table <dest-name> ) | (dest-interface
     <dest-name> ) } [<func> ] ) | none } )
[no] calling-party <name> <key> ({({(dest-table <dest-name> ) | (dest-interface
     <dest-name> ) } [<func> ] ) | none } )
[no] time <name> <key> ({({(dest-table <dest-name> ) | (dest-interface <dest-name> )
     } [<func> ] ) | none } )
[no] date <name> <key> ({({(dest-table <dest-name> ) | (dest-interface <dest-name> )
     } [<func> ] ) | none } )
[no] weekday <name> {sun | mon | tue | wed | thu | fri | sat | default } ({({(dest-
     table <dest-name> ) | (dest-interface <dest-name> ) } [<func> ] ) | none } )
[no] bearer-capability <name> {audio31 | audio71 | rd | speech | ud | video |
     default } ({({(dest-table <dest-name> ) | (dest-interface <dest-name> ) }
     [<func> ] ) | none } )
[no] presentation-indicator <name> {allowed|restricted|interworking|default}
     {({(dest-table <dest-name> ) | (dest-interface <dest-name> ) } [<func> ] ) |
     none }
[no] screening-indicator <name> {default|user-not-screened|user-passed|user-
     failed|network} {({(dest-table <dest-name> ) | (dest-interface <dest-name> ) }
     [<func> ] ) | none }
[no] number-manipulation <name> {cdpn | cnpn } {(add <param> ) | (remove <param> ) |
     (replace <param> ) | (truncate <param> ) | (set-presentation-indicator
     {allowed|restricted|interworking} ) | (set-screening-indicator {user-not-
     screened|user-passed|user-failed|network} ) | (set-type-of-number {unknown |
     national | international | subscriber | network-specific | abbreviated } ) |
     (set-numbering-plan {unknown | e164 | data | telex | national | private } ) |
     convert-to-unknown | convert-to-specific }
[no] complex-function <name> <param>
[no] translation-table <name> <num_in> <num_out>
[no] use tone-set-profile <name>
[no] shutdown
```

interface_pstn

```
[no] interface pstn <if-name>
[no] routing {(dest-table <name>)|(dest-interface <name>) }
[no] replace cnpn
[no] interface pstn <if-name>
[no] routing {(dest-table <name> ) | (dest-interface <name> ) }
[no] fallback {(dest-table <name> ) | (dest-interface <name> ) }
[no] digit-collection {(timeout [<val> ] ) | (terminating-char <val> ) | (nr-length
     <val> ) }
[no] bind port <slot> <port>
[no] use tone-set-profile <name>
exit
```

interface_h323

```
[no] interface h323 <if-name>
[no] bind gateway h323
[no] routing {(dest-table <name> ) | (dest-interface <name> ) }
[no] replace cnpn
[no] fallback {(dest-table <name> ) | (dest-interface <name> ) }
[no] digit-collection {(timeout [<val> ] ) | (terminating-char <val> ) | (nr-length
     <val> ) }
[no] remoteip <remote_ip>
[no] portaddress <portaddress>
[no] codec {g711alaw64k|g711ulaw64k|g723_6k3|g729|transparent|t38_udp} [exclusive ]
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] dtmf-relay
[no] echo-canceller
[no] silence-compression
voice-volume <voice_volume>
[no] use tone-set-profile <name>
exit
```

interface_isoip

```
[no] interface isoip <if-name>
[no] bind gateway isoip
[no] routing {(dest-table <name> ) | (dest-interface <name> ) }
[no] replace cnpn
[no] fallback {(dest-table <name> ) | (dest-interface <name> ) }
[no] digit-collection {(timeout [<val> ] ) | (terminating-char <val> ) | (nr-length
     <val> ) }
[no] remoteip <remote_ip>
[no] portaddress <portaddress>
[no] codec {(transparent [<tx_packet_length> ] ) | (g711alaw64k [<tx_packet_length>
     ] ) | (g711ulaw64k [<tx_packet_length> ] ) | (g723_5k3 [<tx_packet_length> ] )
     | (g723_6k3 [<tx_packet_length> ] ) | (g729 [<tx_packet_length> ] ) | (g726_16k
     [<tx_packet_length> ] ) | (g726_24k [<tx_packet_length> ] ) | (g726_32k
     [<tx_packet_length> ] ) | (g726_40k [<tx_packet_length> ] ) | (g727_16k
     [<tx_packet_length> ] ) | (g727_24k [<tx_packet_length> ] ) | (g727_32k
     [<tx_packet_length> ] ) | (netcoder_6k4 [<tx_packet_length> ] ) | (netcoder_9k6
     [<tx_packet_length> ] ) }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] dtmf-relay
[no] echo-canceller
[no] silence-compression
voice-volume <voice_volume>
[no] use tone-set-profile <name>
exit
exit
```

gateway_isoip

```
gateway isoip [isoip ]
[no] codec {(g711alaw64k [<txlen> ] ) | (g711ulaw64k [<txlen> ] ) | (g723_6k3
     [<txlen> ] ) | (g723_5k3 [<txlen> ] ) | (g729 [<txlen> ] ) | (transparent
     [<txlen> ] ) | (g726_16k [<txlen> ] ) | (g726_24k [<txlen> ] ) | (g726_32k
     [<txlen> ] ) | (g726_40k [<txlen> ] ) | (g727_16k [<txlen> ] ) | (g727_24k
     [<txlen> ] ) | (g727_32k [<txlen> ] ) | (netcoder_6k4 [<txlen> ] ) |
     (netcoder_9k6 } [<txlen> ] ) }[no] shutdown
use voip-profile <profile_name>
exit
```

gateway_h323

```
gateway h323 [h323 ]
[no] alias {h323-id | e164 } <alias>
[no] supported-prefix <prefix> ]
[no] codec {(g711alaw64k [<txlen> <rxlen> ] ) | (g711ulaw64k [<txlen> <rxlen> ] ) |
     (g723_6k3 [<txlen> <rxlen> ] ) | (g729 [<txlen> <rxlen> ] ) | (transparent
     [<txlen> <rxlen> ] ) | (t38_udp ) }
[no] faststart
[no] early-h245
[no]] h245-tunneling
terminal-type {terminal | gateway }
[no] h225-status-enquiry
[no] ras
call-signaling-port <ip_port>
[no] gatekeeper-discovery {(auto [<gkid> ] ) | (manual <ip_address> <ip_port>
     [<gkid> ] ) }
[no] q931-tunneling [isoip-2 isoip-ig ]
[no] clip-clir-support
[no] h235security [(version {v1 } ) | ({encrypted-password | clear-password } <pass-
     word> ) | (master-password <password> ) | (time-window <time-window> ) | (gen-
     eral-id <general-id> ) ]
[no] bind interface <if> [router ]
[no] shutdown
use voip-profile <profile_name>
exit
```

port_ethernet

```
port ethernet <slot> <port>
medium {auto | ({10 | 100 } {half | full } ) }
encapsulation {ip|pppoe|multi}
frame-format {standard|dot1q}
[no] vlan [<vlan_id> ]
cos {(default <default> ) | (rx-map <cos> as <service> ) | (tx-map <service> as
     <cos> ) }
[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_pstn_all

```
port pstn <slot> all
down
```

```
[no] use profile pots <file>
up
exit
```

port_isdn

```
port {pstn | isdn } <slot> <port>
[no] down
[no] channel-range <low> <high>
channel-hunting {up | down | up-cyclic | down-cyclic }
l2proto {pp | pmp }
l3proto {dss1 | pss1 }
channel-numbering {etsi | pss1-old }
[no] loop <channel>
[no] max-channels <channels>
[no] smart-disconnect {from-isdn-calls | to-isdn-calls }
uni-side {net | usr }
clock-mode {master | slave }
[no] permanent-layer2
[no] send ais
[no] pstn-loopback {payload-ts0-regenerated | payload-ts0-transparent | remote-no-
     jitter-attenuator | remote-with-jitter-attenuator }
line-code {ami | hdb3 | b8zs }
[no] caller-id {pre-ring | 1 } [format {bell | etsi } ]
end-of-call signaling {busy tone (loop-break <key_dur> ) } ]
[no] meter-pulses
[no] subscriber-number <key_number>
[no] up
exit
exit
exit
exit
```

RedBoot Bootloader command set

Show help

	Command	Purpose
Step 1	**help** [*topic*]	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 *location* [-l *length*] [-s] [-1|2|4] **x** -b *location* [-l *length*] [-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 *local_ip_address* [/ *mask_len*]] [-h *server*] [-g *gateway*]	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 *count*] [-l *length*] [-t *timeout*] [-r *rate*] [-i *IP_address*] -h *IP_address*	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. I.e. 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 *host*] [-m *various*] [-c *channel*] [-b *base_address*] *file_name*	Loads a program via TFTP, X- or Y-modem (serial port) to memory. -r: Selects raw or ELF data mode. Use –r for Smart-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. file_name: Name of program to load. Used for TFTP transfer.

Execute a program loaded into memory

	Command	Purpose
Step 1	**go** [-w *timeout*] [-i] [-s *script-name*] [*entry*]	Executes a program that has formerly been loaded to memory. The command will check whether a SmartWare 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          SmartWare R2.10 BUILD28015


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
```

Load an image into RAM so that it can be started

	Command	Purpose
Step 1	**fis load** [-b *memory_load_address*] [-n *index*]	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 *mem_base* -l *image_length* [-n *index*]	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 *index*	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 SmartWare to reflect the current SmartWare settings. In other words, it will set the local IP address to the address actually configured in SmartWare.

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	**eeprom read** -e *eeprom_addr* -b *mem_base* -l *length* [-i *device*]	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 C Internetworking terms & acronyms

Chapter contents

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
ISoIP	ISDN over Internet Protocol
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

Abbreviation	Meaning
LT	Line Termination
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 Commit-tee
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

Abbreviation	Meaning
RTM	Route Table Manager
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
SDSL	Symmetric Digital Subscriber Line
SGCP	Simple Gateway Control 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 D Used IP ports & available voice codecs in SmartWare

Chapter contents

Used IP ports in 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)
ISoIP	UDP 1106	Voice data (RTP)
	UDP 1107	Voice statistics (RTCP)
	TCP 1106	Signaling control messages
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 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