PATTON Electronics Co.

IPLink Series

IPLink Software Release 3.20

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: **13220U8-001 Rev. A**
Part Number: **07M2800-SCG**
Revised: **May 3, 2006**

Patton Electronics Company, Inc.
7622 Rickenbacker Drive
Gaithersburg, MD 20879 USA
Tel: +1 (301) 975-1000
Fax: +1 (301) 869-9293
Support: +1 (301) 975-1007
Web: www.patton.com
E-mail: support@patton.com

Trademark Statement

The term *IPLink* is a trademark of Patton Electronics Company. All other trademarks presented in this document are the property of their respective owners.

Notices

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

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

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

Released Build Numbers

This version of the Software Configuration Guide is based on **IPLink 3.20 Build Series 2006-02-02**. The following are the applicable released build numbers:

IPLink 2805 R3.20 Build 2006-03-03	IPLink 2802/2821/2835 R3.20 Build 2006-0-03

Supported Platforms

IPLink Model 2805 (Rev. B)	IPLink 2802/2821/2835 (Rev. D)

Summary Table of Contents

Table of Contents

List of Figures

List of Tables

About this guide

The objective of this *IPLink software Command Configuration Guide* is to provide information concerning the syntax and usage of the command set. For hardware configuration information, refer to the getting started guide that came with your IPLink 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 21)

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

How to read this guide

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

- Appendix A, "Terms and definitions" on page 304 contains the terms and their definitions that are used throughout this *IPLink software Software Configuration Guide*
- Chapter 1, "System overview" on page 24 provides an overview of the main elements of an IPLink system.
- Chapter 10, "IP context overview" on page 110
- Chapter 27, "CS context overview" on page 327

Structure

This guide contains the following chapters and appendices:

- Chapter 1, "System overview" on page 24 provides an overview of the main elements of an IPLink system.
- Chapter 2, "Configuration concepts" on page 29 introduces basic IPLink software configuration concepts.

- Chapter 3, "Command line interface (CLI)" on page 33 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 38 describes the procedures for entering IPLink software 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 50 explains how to establish network-based connections to and from your IPLink by using IP interfaces and Ethernet ports.
- Chapter 6, "System image handling" on page 57 describes how to load and maintain system images and driver software.
- Chapter 7, "Configuration file handling" on page 71 describes how to upload and download configuration files from and to an IPLink devices.
- Chapter 8, "Basic system management" on page 87 describes parameters that report basic system information to the operator or administrator, and their configuration.
- Chapter 9, "RADIUS Client Configuration" on page 99 provides an overview of the authentication, authorization, and accounting (AAA) component in IPLink software and describes how to configure the RADIUS client, a subpart of the AAA component.
- Chapter 10, "IP context overview" on page 110 outlines IPLink software *Internet protocol* (IP) context, together with its related components.
- Chapter 11, "IP interface configuration" on page 117 provides a general overview of IPLink interfaces and describes the tasks involved in their configuration.
- Chapter 12, "NAT/NAPT configuration" on page 128 provides a general overview of the network address port translation and describes the tasks involved in its configuration.
- Chapter 13, "Ethernet port configuration" on page 137 provides an overview of Ethernet ports and describes the tasks involved in their configuration through IPLink software.
- Chapter 14, "Link scheduler configuration" on page 148 describes how to use and configure IPLink software *quality of service* (QoS) features.
- Chapter 15, "Serial port configuration" on page 167 provides an overview of the serial port and describes the tasks involved in its configuration through IPLink software.
- Chapter 16, "T1/E1 port configuration" on page 185 provides an overview of the T1/E1 ports, their characteristics and the tasks involved in the configuration.
- Chapter 17, "Basic IP routing configuration" on page 195 provides an overview of IP routing and describes the tasks involved in configuring static IP routing in IPLink software.
- Chapter 18, "RIP configuration" on page 201 provides an overview of the *routing information protocol* (RIP) and describes the tasks involved in configuring RIP features within IPLink software.
- Chapter 19, "Access control list configuration" on page 211 provides an overview of IP access control lists and describes the tasks involved in their configuration through IPLink software.
- Chapter 20, "SNMP configuration" on page 225 provides overview information about the *simple network management protocol* (SNMP) and describes the tasks used to configure those of its features supported by IPLink software.

- Chapter 21, "SNTP client configuration" on page 240 describes how to configure a *simple network time protocol* (SNTP) client.
- Chapter 22, "DHCP configuration" on page 251 provides an overview of the *dynamic host configuration control protocol* (DHCP) and describes the tasks involved in its configuration.
- Chapter 23, "DNS configuration" on page 261 describes how to configure the *domain name system* (DNS) component.
- Chapter 24, "DynDNS configuration" on page 265 describes configuring the *dynamic DNS* (DynDNS) service.
- Chapter 25, "PPP configuration" on page 270 describes how to configure the *point-to-point protocol* over different link layers.
- Chapter 26, "VPN configuration" on page 287 describes how to configure the VPN connections between two IPLink devices or between an IPLink and a third-party device.
- Chapter 27, "CS context overview" on page 327 gives an overview of IPLink software *circuit-switching* (CS) context and its associated components and describes the tasks involved in its configuration.
- Chapter 28, "CS interface configuration" on page 349 gives an overview of interfaces in the CS context and describes the tasks involved its configuration.
- Appendix A, "Terms and definitions" on page 304 contains the terms and their definitions that are used throughout this *IPLink software Software Configuration Guide*.
- Appendix B, "Mode summary" on page 309 illustrates the modes hierarchy.
- Appendix C, "Command summary" on page 313 is a command reference.
- Appendix D, "Internetworking terms & acronyms" on page 325 contains terms and definitions relating to internetworking.
- Appendix E, "Used IP ports in the IPLink software" on page 330 describes the used IP ports and available voice codecs in IPLink software.

Precautions

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

Note A note presents additional information or interesting sidelights.

IMPORTANT

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

Typographical conventions used in this document

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

General conventions

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

Table 1. General conventions

Convention	Meaning
Garamond blue type	Indicates a cross-reference hyperlink that points to a figure, graphic, table, or section heading. Clicking on the hyperlink jumps you to the reference. When you have finished reviewing the reference, click on the **Go to Previous View** button ◀ in the Adobe® Acrobat® Reader toolbar to return to your starting point.
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
Garamond italic type	Indicates the names of fields or windows.
Garamond bold type	Indicates the names of command buttons that execute an action.
< >	Angle brackets indicate function and keyboard keys, such as **<shift>**, **<ctrl>**, **<c>**, and so on.
[ ]	Elements in square brackets are optional.
{a | b | c}	Alternative but required keywords are grouped in braces ({ }) and are separated by vertical bars (|)
node	The leading IP address or nodename of an IPLink is substituted with ***node*** in ***boldface italic*** font.
IPLink	The leading **IPLink** on a command line represents the nodename of the IPLink
#	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 and support

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

Patton support headquarters in the USA

- Online support: Available at **www.patton.com**
- E-mail support: E-mail sent to **support@patton.com** will be answered within 1 business day
- Telephone support: Standard telephone support is available five days a week—from **8:00 am** to **5:00 pm** EST (1300 to 2200 UTC/GMT)—by calling +1 (301) 975-1007
- Support via VoIP: Contact Patton free of charge by using a VoIP ISP phone to call **sip:support@patton.com**
- Fax: +1 (253) 663-5693

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

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

Warranty Service and Returned Merchandise Authorizations (RMAs)

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

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

Warranty coverage

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

Returns for credit

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

Return for credit policy

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

RMA numbers

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

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

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

Shipping instructions

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

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

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

Chapter 1 System overview

Chapter contents

Introduction

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

- IPLink hardware platforms (see page 26)
- IPLink software embedded software (see page 26)

A complete IPLink system or network is typically composed of the following main elements plus a third-party network infrastructure (see figure 1):

- The first and most obvious element is the *IPLink* devices (also referred to as *hardware platforms* or *network nodes*) that provide the physical connectivity and the CPU resources. All IPLink models support packet-routed traffic.
- The second element comprises the embedded software—called *IPLink software*—running on the IPLink hardware platforms.
- Finally, a third-party IP network and transmission infrastructure provides IP connectivity between the above elements. This infrastructure can range from a simple Ethernet hub or switch to highly complex networks including multiple access technologies, backbone transmission, and services nodes.

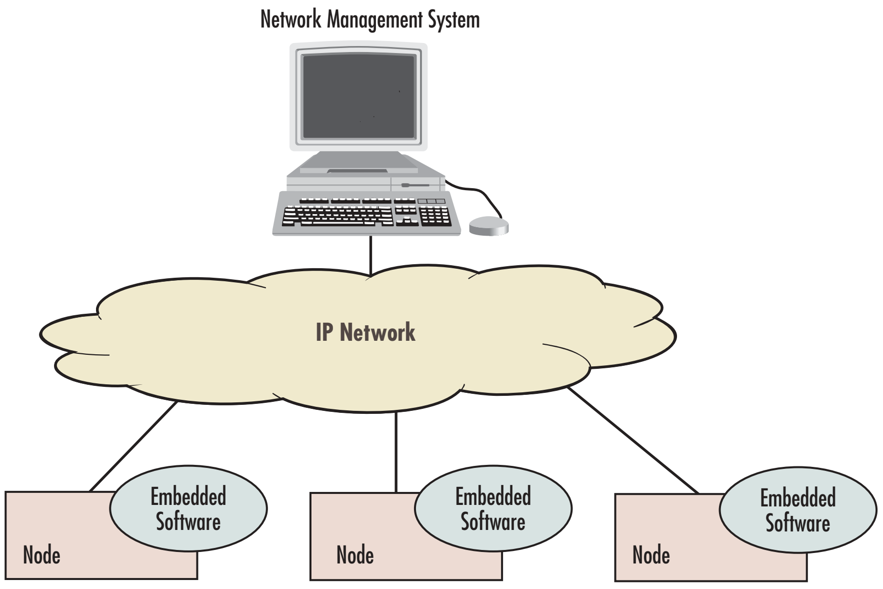


Figure 1. System overview with a Management Center

IPLink hardware platforms

The IPLink series of devices covers a performance range varying from that suitable for small office/home office (SOHO) applications to enterprise and carrier sites. Table 3 lists the IPLink models.

Table 3. IPLink Products

Model	Description	Ethernet Ports	WAN/Data Interfaces
2802	Managed VPN Router with WAN and LAN Ethernet ports	2	—
2803	Managed Serial VPN Router with T1/E1 WAN port	2	T1/E1
2805	Managed VPN Router with WAN (1) and LAN (4) Ethernet ports	5	—
2821	Managed Serial VPN Router with X.21 WAN port	2	X.21
2823	Managed VPN Router with WAN, LAN, and DMZ Ethernet ports	3	—
2835	Managed Serial VPN Router with V.35 WAN port	2	V.35

Figure 2 depicts the basic system model of a Patton IPLink. All IPLink devices 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 IPLink system.

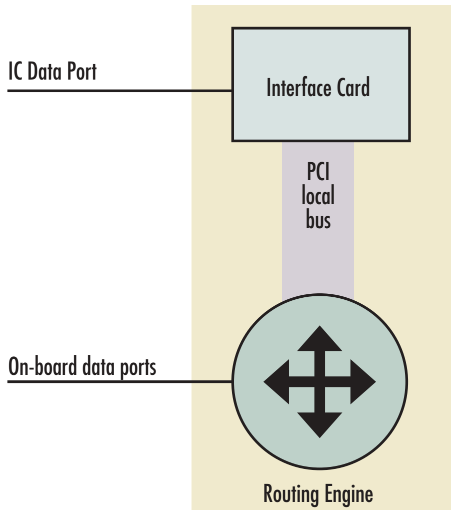


Figure 2. IPLink system model

IPLink software embedded software

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

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

An IPLink software build is a binary image file. It is usually divided into several checksum-protected files to improve download efficiency and security. The download to the IPLink is handled in sequence by using a download *batchfile*. Refer to chapter 6, "System image handling" on page 57 for details on IPLink software image downloads.

In addition to the actual IPLink software 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 IPLink software application. It also provides minimal network services, allowing the IPLink to be accessed and upgraded over the network even if IPLink software application should not start. The boot loader is installed in the factory and requires no upgrading.
- The *PMC loader* initializes the PMC interface cards when mounted in IPLink devices. It checks the hardware versions and determines whether compatible PMC drivers are available. The PMC loader may be upgraded together with an IPLink software release.
- The *PMC driver* software performs the runtime tasks on the PMC interface cards mounted in IPLink devices. The PMC drivers are interface card specific and also have build numbers. Refer to the IPLink software release notes for PMC driver software compatibility. The PMC drivers may be upgraded together with an IPLink software release or they can be downloaded individually onto the device's flash memory file system.

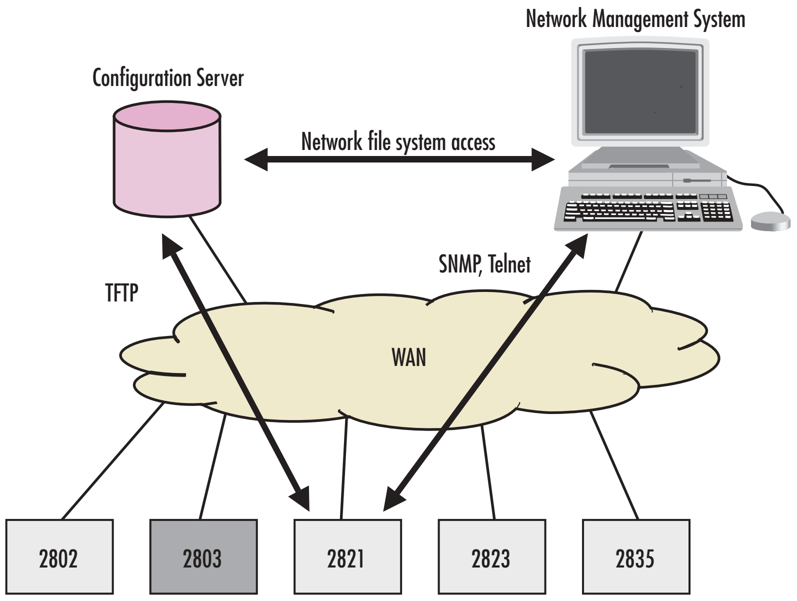


Figure 3. IPLink Management System

IPLink Software management center tools

IPLink software 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 IPLink devices 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 IPLink management into standard network management platforms such as HP OpenView.

Chapter 2 Configuration concepts

Chapter contents

Introduction

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

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

This chapter includes the following sections:

- Contexts (see page 31)
- Interfaces, ports, and bindings (see page 31)
- Profiles and Use commands (see page 32)

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

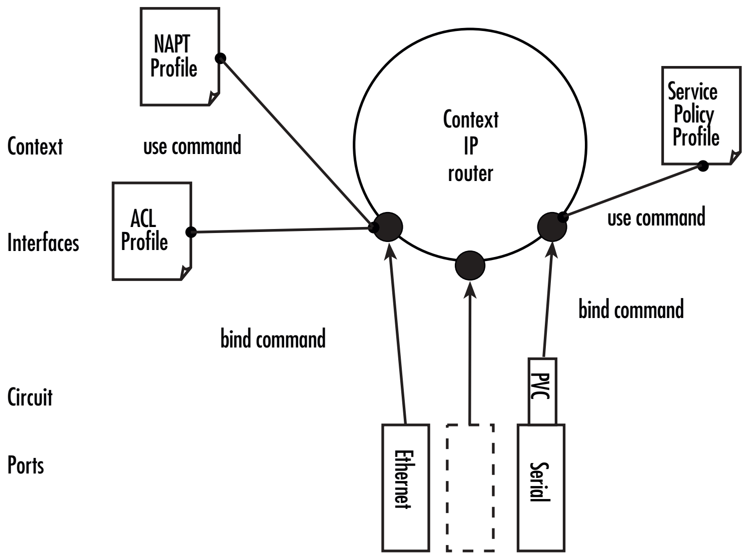


Figure 4. Configuration concept overview

Figure 4 shows the various elements of a complete IPLink configuration. Each of these elements implements one of the configuration concepts described in this chapter. The figure also shows the relationships and associations between the different elements. The relations are specified through *bind* (arrow) and *use* (bullet-lines) commands. For example, you need *bind* commands to bind a physical port to a logical interface, and *use* commands to assign profiles to contexts.

The sections that follow refer to figure 4 on page 30 and describe the concepts and elements in more detail.

Contexts and Gateways

Context

An IPLink software *context* represents one specific networking technology or protocol, namely IP (Internet Protocol). A context can be seen as *virtual dedicated equipment* within the IPLink. For example:

- An IP context contains the routing functions of the IPLink. It can be thought of as an embedded router within the IPLink

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

Each context contains a number of *interfaces*, which build the connections to other IPLink software elements and the outside world. Figure 4 on page 30 shows one context:

- one of type IP named *router*

This corresponds to the default configuration of all IPLink devices.

Note IPLink software currently supports only one instance of the IP context types.

Example

The IP context named *router* can contain static routes, RIP, and NAT configuration parameters.

Interfaces, Ports, and Bindings

Interfaces

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

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 IPLink software. Refer to the “Bindings” section for more information. In figure 4 on page 30, the IP context shows three interfaces. These interfaces are configured within their context. 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 IPLink software represent the physical connectors and channels on the IPLink hardware. The configuration of a port or circuit includes parameters for the physical and data link layer such as line clocking, line code, framing and encapsulation formats or media access control. Before any higher-layer user data can flow through a physical port or circuit, you must associate that port or circuit with an interface on a context. This association is referred to as a *binding*. Refer to the “Bindings” section for more information.

Examples of IPLink ports are: 10Base-T Ethernet, Serial T1/E1, V.35, and X.21. Ports are numbered according to the IPLink port numbering scheme. The port name corresponds to the label (or abbreviation) printed on the hardware.

Example: Ethernet 0/1, Serial 0/0

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

Example: frame-relay pvc 112.

Figure 4 on page 30 shows three ports. Three ports are bound directly to an IP interface. One port has a single circuit configured, which is bound to the IP context.

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.

Bindings from ports to IP interfaces are shown in figure 4 on page 30.

Profiles and Use commands

Profiles

Profiles provide configuration shortcuts. They contain specific settings that can be used in multiple contexts or interfaces. 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 are shown in figure 4 on page 30.

Use Commands

Use commands form the association between profiles and contexts, 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 IPLink software is the command line interface (CLI). You can access the CLI via the IPLink console port or through a Telnet session. The CLI lets you configure the complete IPLink software functionality, as opposed to the SNMP and HTTP management interfaces that offer a more limited subset of the functions. You can enter CLI commands online or as a configuration script in the form of a text file. The CLI also includes monitoring and debugging commands. CLI commands are simple strings of keywords and user-specified arguments.

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

- Command Modes
- Command Editing (see page 35)

Command modes

The CLI is composed of modes. There are two *mode groups*: the *exec mode* group and the *configuration mode* group. Within the exec mode group there are two modes: *operator exec* and *administrator exec*. The configuration mode group contains all of the remaining modes. A command mode is an environment within which a group of related commands is valid. All commands are mode-specific, and certain commands are valid in more than one mode. A command mode provides command line completion and context help within the mode. The command modes are organized hierarchically. The current working mode is indicated by the CLI prompt. Appendix B, "Mode summary" on page 309 contains a detailed overview of all command modes, and appendix C, "Command summary" on page 313 describes the commands that are valid in each mode.

CLI prompt

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

```
nodename>
```

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

```
nodename#
```

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

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

Where:

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

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

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

The CLI commands used to enter each mode and the system prompt that is displayed when you are working in each mode is summarized in appendix B, "Mode summary" on page 309.

Navigating the CLI

Initial mode

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

System changes

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

Configuration

To make configuration changes, the configuration mode must be entered by using the **configure** command in the administrator exec mode. After doing that, other configuration modes are accessible, as diagrammed in the overview in figure 4 on page 30.

Changing Modes

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

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

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

Command editing

Command help

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

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

The No form

Almost every command supports the keyword **no**. Typing the **no** keyword in front of a command disables the function or "deletes" a command from the configuration. For example, to enable the Session Router trace tool, enter the command **debug session-router**. To subsequently disable the Session Router trace, enter the command **no debug session-router**.

Command defaults—returning parameters to default values

The parameters of certain commands are set to their default value simply by omitting the parameter. For example:

- **sntp-client local-port 220**—Sets the UDP port used by the SNTP client to *220*
- **sntp-client local-port**—Sets the UDP port used by the SNTP client back to its default setting of *123*

The other examples of such commands are:

sntp-client operating-mode	**sntp-client anycast-address**	**sntp-client local-port**
sntp-client poll-interval	**penalty-box-time**	**sntp-client gmt-offset**
server-timeout	**timeout**	**domain**

Command completion

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

Command history

IPLink software maintains a list of previously entered commands that you can go through by pressing the **<up-arrow>** and **<down-arrow>** keys, and then pressing **<enter>** to enter the command.

The show history command displays a list of the commands you can go through by using the arrow keys.

Command Editing Shortcuts

IPLink software 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 36. The syntax **<Ctrl>-<p>** means press the **<p>** key while holding down the keyboard's control key (sometimes labeled *Control, Ctl*, or *Ctrl*, depending on the keyboard and operating system of your computer).

<Esc>-<f> is handled differently; press and release the escape key (often labeled *Esc* on many keyboards) and then press the **<f>** key.

Table 4. Command edit shortcuts

Keyboard	Description
<Ctrl>-<p> or **<up-arrow>**	Recall previous command in the command history.
<Ctrl>-<p> or **<up-arrow>**	Recall next command in the command history.
<Ctrl>-<p> or **<up-arrow>**	Move cursor forward one character.
<Ctrl>-<p> or **<up-arrow>**	Move cursor backward one character.

Table 4. Command edit shortcuts (Continued)

Keyboard	Description
<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.
<Ctrl>-<v>	Insert a code to indicate to the system that the keystroke immediately following should be treated as normal text, not a CLI command. For example, pressing the question mark **<?>** character in the CLI prints a list of possible tokens. If you want to use the ? in a configuration command, e.g. to enter a regular expression, press **Ctrl-v** immediately followed by the question mark <**?**>.

Chapter 4 Accessing the CLI

Chapter contents

Introduction

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

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

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

The ports available for connection and their labels for each IPLink model are shown in the getting started guide that came with your IPLink 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 IPLink software 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 33. The telnet server can be disabled if desired.

IMPORTANT

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

Accessing the IPLink software CLI task list

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

- Accessing via the console port (see page 40)
- Accessing via a Telnet session (see page 41)
- Using an alternate TCP listening port for the Telnet server (see page 42)
- Disabling the Telnet server (see page 42)
- Logging on to the IPLink software (see page 42)
- Selecting a secure password (see page 43)
- Configuring operators and administrators (see page 43)
- Displaying the CLI version (see page 45)
- Displaying account information (see page 46)
- Switching to another log-in account (see page 46)
- Checking identity and connected users (see page 47)
- Ending a Telnet or console port session (see page 49)

Accessing via the console port

To access an IPLink via its console port, the host computer must be connected directly to the console port (labeled CONSOLE) with a serial cable (see figure 5). The host must use a terminal emulation application that supports serial interface communication.

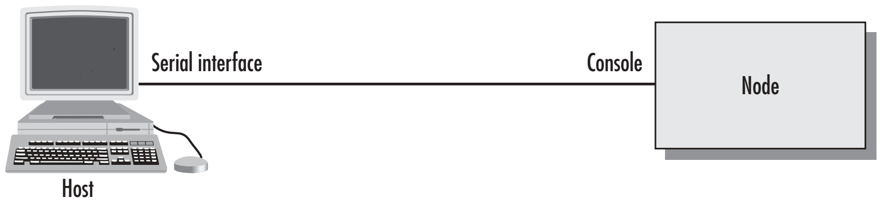


Figure 5. Setup for initial configuration via the console port

Note You do not need to configure IP settings if you access the IPLink 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 IPLink 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 IPLink. Use a serial cable according to *Appendix A* of the getting started guide included with your IPLink device.
4. Power on your IPLink. 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 an IPLink. The Telnet host accesses the IPLink via its network interface. A host can be connected directly to the ETH 1 port (LAN) with a crossover cable (see figure 6, part A) or through an Ethernet hub with two straight cables (see figure 6, part B).

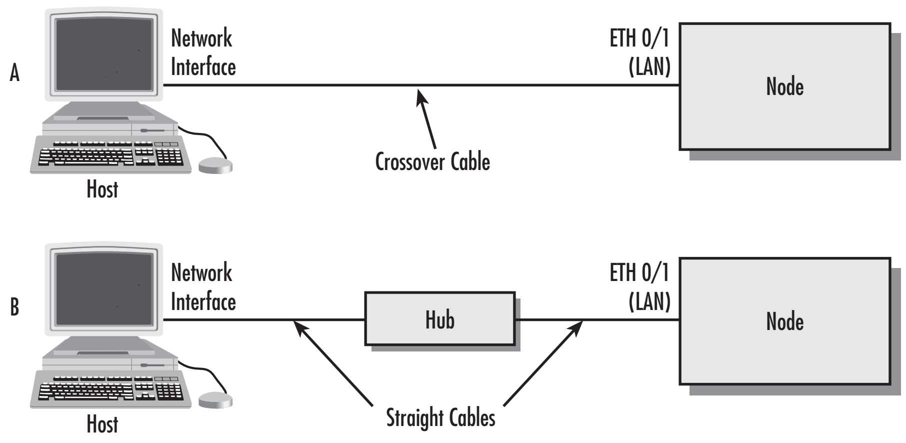


Figure 6. Setup for initial configuration via an Ethernet port

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 IPLink. Table 5 lists the default IP address and network mask of the Ethernet ports of the IPLink.

Table 5. Factory default IP address and network mask configuration

Item	IP Address	Network Mask
WAN interface Ethernet 0 (ETH 0/0)	DHCP client	DHCP client
LAN interface Ethernet 1 (ETH 0/1)	192.168.1.1	255.255.255.0
DHCP server address range	192.168.1.10–192.168.1.99	255.255.255.0

Note The DHCP server is running on the ETH 0/1. All Ethernet ports are pre-configured and active.

Note The default IP addresses listed in table 5 apply to an operating scenario compatible with the factory configured settings of the IPLink. If your operating requirements are significantly different, your IPLink may have different default IP addresses. Check IPLink software release notes for more details.

Telnet Procedure

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

1. Set up the IPLink as described in the Quick Start Guide included with your IPLink device.

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

3. Power on your IPLink and wait until the *Run* LED lights.
4. Set your PC is set to DHCP.
5. Open a Telnet session to the ETH 1 (LAN) port with the IP address 192.168.1.1 of your IPLink.
6. Proceed with logging in.

Using an alternate TCP listening port for the Telnet server

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

Mode: Configure

Step	Command	Purpose
1	[*name*]**(cfg)# terminal telnet port <port>**	Uses TCP port <port> for accepting telnet connections

Disabling the Telnet server

The telnet server can be disabled using the following command.

Mode: Configure

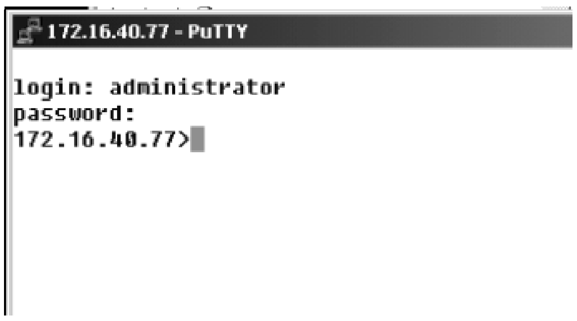
Step	Command	Purpose
1	[*name*]**(cfg)# no terminal telnet**	Disables the telnet server

Logging onto the IPLink software

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

The opening Telnet screen you see resembles that shown in figure 7. The window header bar shows the IP address of the target IPLink.

A factory preset administrator account with name *administrator* and an empty password is programmed into the IPLink software 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 7. 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 7, such as the IP address in the system prompt and window header bar, may be different on your IPLink 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 applies to you:

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

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

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

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

Password encryption

Password encryption is the same for all passwords: administrator/operator accounts, PPP, DynDNS, and so on). You can enter the password in two formats:

- Plain text, for example: *secretpassword*
- Encrypted, for example: *HUAvCYeILWZz33hQvS0IEpQ==encrypted*

When executing the command **show running-config**, the passwords are always displayed in encrypted format. A password is encrypted by entering it in plain text, then executing the **show running-config** command. You may also store it permanently into the startup-config by using the command **copy running-config startup-config**.

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 IPLink software via the console or through Telnet.

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

Password encryption

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

Note The password encryption feature prevents unauthorized users from getting passwords by looking at the configuration file. Other types of encryption that can be configured in the IPLink software (ESP, VPN, and IPsec) protect data transmitted from the IPLink device.

The procedure for password encryption is the same for all the passwords (administrator/operator accounts, PPP, DynDNS, RADIUS). Passwords can be entered in as follows:

- Plain text
- Encrypted text (for example, the password mypassword always appears in encrypted form as *HUAvCYeILWZz3hQvS0IEpQ== encrypted* when doing a **show** command)

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

Factory preset administrator account

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

Creating an operator account

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

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

Mode: Operator execution

Step	Command	Purpose
1	***node*>enable**	Enters administration execution mode
2	***node*#configure**	Enters configuration mode

Step	Command	Purpose
3	***node(cfg)#* operator** *name* **password** *password*	Creates a new operator account *name* and password *password*
4	**copy running-config startup-config**	Saves the change made to the running configuration of the IPLink, so that it will be used following a reload

Example: Create an operator account

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

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

Creating an administrator account

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

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

Mode: Operator execution

Step	Command	Purpose
1	***node*>enable**	Enters administration execution mode
2	***node*#configure**	Enters configuration mode
3	***node(cfg)#* administrator** *name* **password** *password*	Creates a new administrator account *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 how to add a new administrator account with a login name *super* and a matching password *Gh3*Ke4h*.

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

Displaying the CLI version

This procedure displays the version of the currently running IPLink software CLI.

Mode: Operator execution

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

Example: Displaying the CLI version

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

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

Displaying account information

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

The following procedure describes how to display account information:

Mode: Administrator execution

Step	Command	Purpose
1	***node*#show accounts**	Displays the currently-configured administrator and operator accounts

Example: Display account information

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

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

Switching to another account

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

Mode: Administrator or operator execution

Step	Command	Purpose
1	***node*>su account-name**	Changes to the user account *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 the operator execution mode. In the example below the **who** command is used to check the identity within both accounts

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

Checking identity and connected users

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

Mode: Administrator or operator execution

Step	Command	Purpose
1	***node#who***	Shows more detailed information about the users ID, name, state, idle time and location
	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 in which you are working.

Used in administrator execution mode:

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

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

Used in operator execution mode:

```
IPLink>who
You are operator support
```

Command index numbers

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

```
192.168.1.1(pf-prov)[testpro]#show running-config
```

```
...
profile provisioning testpro
  location 1 tftp://10.10.1.2/test1.cfg
  location 2 tftp://10.10.1.2/test2.cfg
  location 3 tftp://10.10.1.2/test3.cfg
...
```

The following command has index numbers:

- **location**

These commands always have index numbers in the running-config. However, entering the index is optional. If you enter such a command with an index, it is inserted into list at the position defined by the index. If you enter such a command without an index, it is placed at the bottom of the list. Also, you can change a commands position in a listing (moving it up or down in the list) by changing its index number.

Example 1: Moving the test1.cfg from position *1* in the list to position *3.*

Listing before changing the location index number:

```
profile provisioning testpro
  location 1 tftp://10.10.1.2/test1.cfg
  location 2 tftp://10.10.1.2/test2.cfg
  location 3 tftp://10.10.1.2/test3.cfg
...
```

Listing after changing index number:

```
192.168.1.1(pf-prov)[testpro]#location 1 after 3
192.168.1.1(pf-prov)[testpro]#show running-config
...
profile provisioning testpro
  location 1 tftp://10.10.1.2/test2.cfg
  location 2 tftp://10.10.1.2/test3.cfg
  location 3 tftp://10.10.1.2/test1.cfg
...
```

Note The IPLink software automatically renumbered the succeeding indexes after location 1 became location 3.

Ending a Telnet or console port session

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

Mode: Operator execution

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

Example: End a Telnet or console port session

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

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

After confirming the dialog with “yes”, the Telnet session to the IPLink 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.

Hidden commands in .cfg configuration files

The command **cli config defaults** turns it on, **no cli config defaults** turns it off. It is turned off by default.

When enabled, commands that are normally hidden will be displayed. Typically these commands comprise well-known standard default values, which is why they are hidden by default to reduce the clutter in a displayed configuration file.

Chapter 5 Establishing basic IP connectivity

Chapter contents

Introduction

This chapter explains how to establish network-based connections to and from your IPLink using IP interfaces and Ethernet ports. You can configure basic IP connectivity in the *context IP* and the subsidiary *interface* command modes. For a complete description of the IP context and interface configuration related commands referred to in this chapter, see chapter 10, "IP context overview" on page 110, and chapter 11, "IP interface configuration" on page 117.

The chapter includes the following sections:

- IP context selection and basic interface configuration tasks
- Examples (see page 55)

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

IP context selection and basic interface configuration tasks

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

- Entering the IP context, creating IP interfaces and assigning an IP address
- Defining IP Ethernet encapsulation and binding an IP interface to a physical port (see page 52)
- Activating the physical port (see page 52)
- Displaying IP interface information (see page 53)
- Deleting IP interfaces (see page 54)

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

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

IPLink software application software running on your IPLink 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 the 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.

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

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

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

Mode: Configure

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

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

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

Activating a physical port

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

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

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

Ethernet Configuration
-------------------------------------

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

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

Step	Command	Purpose
1	***node*(ctx-ip)[router]#port ethernet** *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

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

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

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

Displaying IP interface information

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

Mode: Configure

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

Example: List existing IP interfaces

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

```
IPLink(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.

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

Deleting IP interfaces

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

Mode: Context IP

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

Example: Delete IP interfaces

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

1. List the existing interfaces:

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

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

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

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

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

Examples

Setting up an IP interface on an Ethernet port

The following example shows all required configuration steps, which end in an activated IP interface on Ethernet port 0 on slot 0. Figure 8 shows the relation between the IP interface *lan* and the Ethernet port 0 on slot 0. The configuration procedure below starts in the operator execution mode:

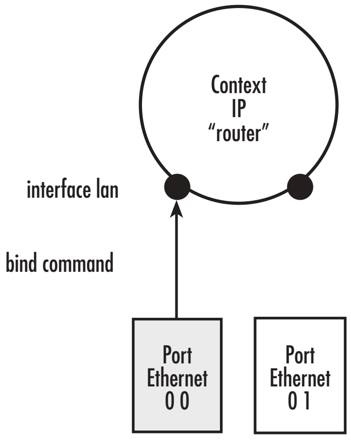


Figure 8. Relation between IP Interface *lan* and Ethernet port 0 on slot 0

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

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

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

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

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

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

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

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

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

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

Chapter 6 System image handling

Chapter contents

Introduction

This chapter describes how to load, maintain, and update the various software images in the IPLink. The IPLink software system software consists of the application image and the driver images. The images are stored in persistent (non-volatile) memory. The application image is the software which actually operates the IPLink. Driver images are used to operate the various optional PMC interface cards.

This chapter includes the following sections:

- Memory regions in IPLink software
- System image handling task list (see page 60)
- Boot procedure and bootloader (see page 65)

Note Section "System image handling task list" on page 60 describes the standard way to upgrade the IPLink software. If you encounter problems that won't let you upgrade using the standard method, refer to section "Bootloader" on page 67 as appropriate.

- Factory configuration (see page 70)

Patton IPLink devices are shipped with default system software which is stored in persistent memory. Along with the default system software (application image and driver images), a factory configuration, *factory-config*, has been loaded into the IPLink at the factory. This configuration file sets the initial basic operating parameters of the IPLink, such as enabling the Ethernet ports, setting the default IP addresses and the DHCP server.

Other configuration files may be stored in the IPLink persistent memory. A configuration file is an ordered list of commands. Some of the various configuration files are

- factory-config (read-only)
- startup-config
- running-config
- user-config1, user-config2, etc. (these are specific application configurations created by the user)

Backups of the configuration files can be stored on a remote *trivial file transfer protocol* (TFTP) server. The remote tftp server must be accessible via one of the IPLink IP interfaces. Tftp cannot be used from the console interface.

The following sections focus on IPLink software memory regions, as well as the software components you can copy into the memory or move between a TFTP server and the memory of the IPLink. As IPLink software uses a specific vocabulary in naming those software components, refer to appendix A, "Terms and definitions" on 304 to ensure that you understand the concepts.

Memory regions in IPLink software

The IPLink's memory contains several logical regions and several physical regions as shown in figure 9 on page 60, each separate from the other.

Note You will use a remote TFTP server for uploading and downloading the application image, the driver images, and the various configuration files to the IPLink. The command syntax in IPLink software requires you to prefix the file path on the TFTP server with *tftp:* followed by the absolute file path. You need to start from the root directory of the TFTP server.

The three physical regions of memory are the remote tftp server's memory, the *Volatile* memories, and the *Persistent* memory in the IPLink. The remote tftp server has one logical region, *tftp:*, which can contain various configuration files and batch files for system software upgrade/download. Within the IPLink the *Volatile* physical region contains one logical region, *system:*, which is random access memory (RAM). When no power is applied to the IPLink, the *system:* region contains no data, no configuration—nothing; it is volatile. The system: region contains the current running configuration, called *running-config*.

The third and last physical memory region is the *Persistent* portion. It has two logical regions called *flash:* and *nvram:*.

- The logical region *flash:* stores the application image, the driver images and the bootloader image. These images are not lost when the IPLink is powered off.
- The logical region *nvram:* stores the various configuration files. The factory default configuration file is always present in *nvram:*, and can be restored as the running-config by pressing the reset button. For those models that do not have a reset button, use the **copy** command. The startup-config and user-specific configurations are also stored in *nvram:*.

The factory configuration is read-only. It is contained in the logical region *nvram:* of the IPLink. It is used—if no user-specific configuration is available—to start-up IPLink software with a minimal functionality. This configuration is named *factory-config* in IPLink software terminology.

On powering up an IPLink (or pressing the Reset button on applicable units) with no pre-configured user configuration files, the default *factory-config* file is also the *startup-config* and the *running-config*. Upon changing any configuration parameters, the changes are made to the running-config in the *system:* region of the Volatile memory. Unless these changes are copied into startup-config or another user-named configuration file, all configuration changes will be lost if the IPLink is powered down.

A dedicated user-specific configuration must be created and stored in the nvram: region of persistent memory. In fact, you may create numerous user-specific configurations in the same IPLink, but if only one dedicated user-specific config is required, you may save it in *startup-config* by using the **copy running-config startup-config** command. Any future time you restart the IPLink, it will use this saved configuration. In other words, the *startup-config* configuration file becomes your default operating configuration.

If you have created and saved numerous user-defined operating configuration files, you can change the startup default configuration file simply by copying the selected config file into *startup-config* and rebooting the IPLink.

Any configuration stored in logical region *nvram:* or *system:* can be copied to a remote server by using TFTP.

Operating configurations cannot be executed from the persistent memory, so the configuration used for operating the IPLink is copied into the volatile memory of the IPLink prior to normal operation. This procedure takes place after the system bootstrap, where the application image (i.e. IPLink software) is started and a configuration must be available. Shortly before IPLink software has completed all startup processes, the configuration *startup-config* is copied from *nvram:* in persistent memory to the *running-config* configuration in system: in volatile memory.

You can back up the *running-config* to *nvram:* or to a remote TFTP server with a user-defined name.

Note When returning to the *factory-config* by using the **copy factory-config startup-config** command, all user-specific configurations saved in *nvram:* remain even after *reload*.

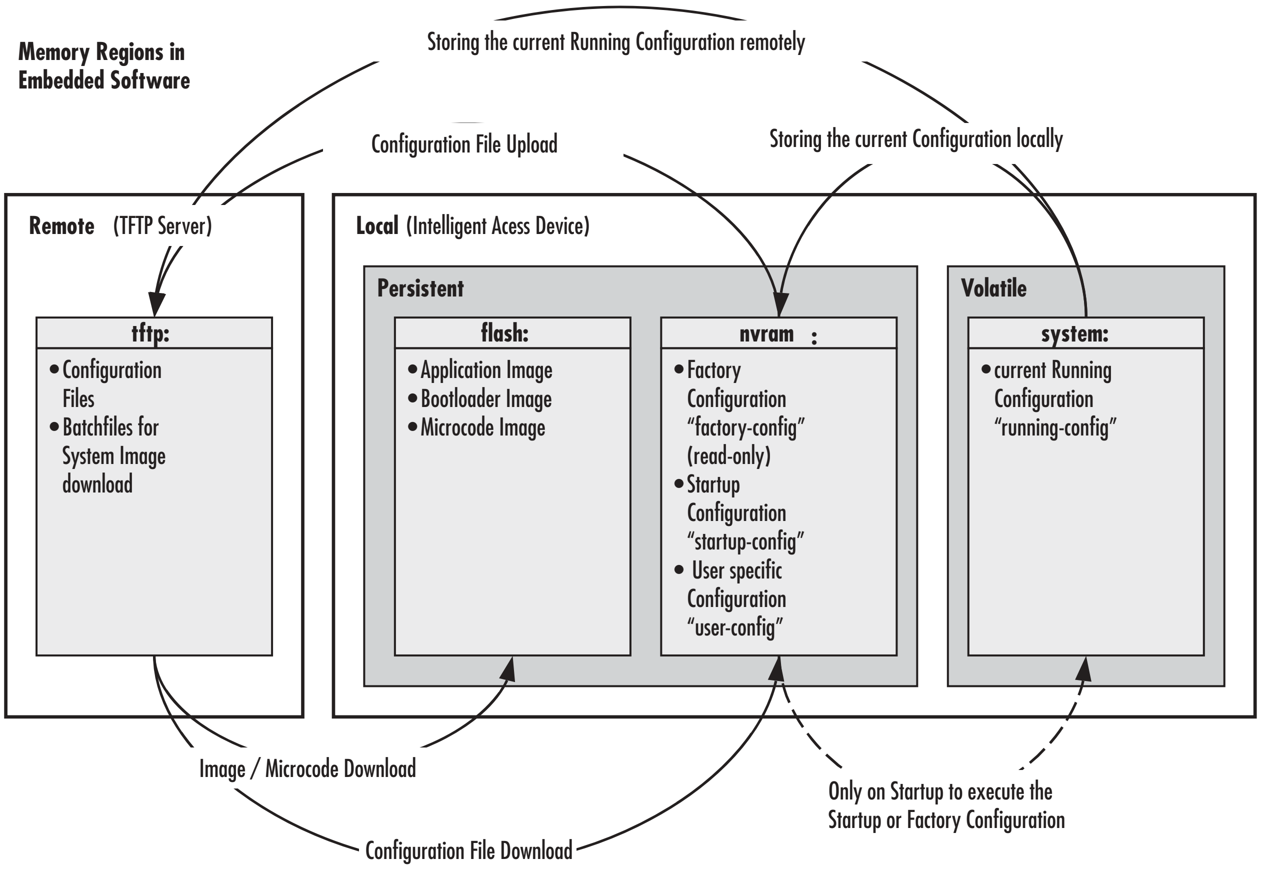

Memory Regions in Embedded Software
Storing the current Running Configuration remotely
Configuration File Upload
Storing the current Configuration locally
Remote (TFTP Server)
tftp:
• Configuration Files
• Batchfiles for System Image download
Local (Intelligent Acess Device)
Persistent
flash:
• Application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

Figure 9. IPLink memory regions logically defined in IPLink software

System image handling task list

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

- Displaying system image information
- Copying system images from a network server to the Flash memory (see page 61)
- Copying the driver software from a network server to the Flash memory (see page 62)

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 that is the currently running system software. If you have just completed a download of new system software from the tftp server, you must execute the **reload** command in order to be running with the new system software. This applies equally to driver software.

Example: Display system image information

The following example shows the information that is available for an IPLink 2800 series device with an optional IC-4BRV interface card mounted in slot 2.

```
IPLink#show version

Product name     : SN2803
Software Version : IPLink software R3.20 2006-02-02 H323 SIP FXS FXO
Supplier         :
Provider         :
Subscriber       :

Information for Slot 0:
2803/K/EUI (Admin State: Application Started, Real State: Application Started)
Hardware Version : 4, 2
Serial number    : 00A0BA01C47C
PLD Version      : 0x00040103
Software Version : IPLink software R3.20 2006-02-02 H323 SIP FXS FXO
```

Copying system images from a network server to Flash memory

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

Since the system image file is preloaded at the Patton Electronics Co. factory, you will have to download a new IPLink software application software only if a major software upgrade is necessary or if recommended by Patton Electronics Co. Under normal circumstances, downloading a system image file should not be needed.

Downloading a new system image file means storing it permanently at a defined location within the IPLink flash memory. To store the system image file, you must use a special download script file. This script file defines how to handle the system image file and where to store it. You cannot download any system image file without an appropriate script file.

Each line in the script file is a command for the CLI of your IPLink. To download a system image file, which will replace the currently running IPLink software 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. 2006-02-02
image.bin 1369474 21; ver 2803.1,2803.2;
cli.xml
+/flash/cli/spec.xml
*UÊDä
```

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

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

Mode: Administrator execution

Step	Command	Purpose
1	***node*(cfg)# copy tftp://***_node-ip-address_**/**_b_ **flash:**	Downloads the script file *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 the Flash memory

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

```
IPLink>enable
IPLink#configure
IPLink(cfg)#copy tftp://172.16.36.80/2803/build22032/b flash:
Completed image download
Completed file download /flash/cli/spec.xml

IPLink(cfg)#
```

After the successful download, issue the **reload** command (in order to start the IPLink with the new software).

Copying driver software from a network server to Flash memory

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

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

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

```
# script file for driver software image download
# Patton Electronics Co. 2006-02-02
;
/Vx_R3.20_BUILD24028
+/flash/bin/pmc000216a6
4_—-
```

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

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

Mode: Administrator execution

Step	Command	Purpose
1	***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 the Flash memory

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

```
IPLink>enable
IPLink#configure
IPLink(cfg)#copy tftp://172.16.36.80/build24028/b flash:
Completed file download /flash/bin/pmc000216a6

IPLink(cfg)#
```

Note In order for the newly downloaded driver software to be running, you must execute the **reload** command.

Auto provisioning of firmware and configuration

The new auto provisioning capability enables you to automatically distribute up-to-date configurations and firmware to a large number of units using TFTP. It works as follows:

The unit downloads a specific file from a TFTP server. If this file has changed since the last download, it is stored and executed. If the file on the server did not change since the last download, no action is taken. If the units are configured to do auto provisioning, a network operator can only update the firmware files on the TFTP server, which automatically distributes it to all units. The "profile provisioning" configures this. Here's an example for firmware provisioning:

```
profile provisioning FIRMWARE
destination script
location 1 tftp://172.16.1.2/firmware/b
location 2 tftp://172.16.1.33/firmware/b
activation reload graceful
```

Explanation:

Step	Command	Purpose
1	[name] **(pf-prov)**[FIRMWARE]**#destination script**	Chooses the unit's script interpreter as destination of the downloaded file. Use this for firmware updates. Script files are the *b, b1, …* files that come with each unit firmware update.
2	[name] **(pf-prov)**[FIRMWARE]**#location 1 tftp://172.16.1.2/firmware/b**	Specifies the location of the file to check for changes.
3	[name] **(pf-prov)**[FIRMWARE]**#location 2 tftp://172.16.1.33/firmware/b**	Specifies alternate locations of the file. If the first could not be contacted, the second is tried, and so on.
4	[name] **(pf-prov)**[FIRMWARE]**#activation reload graceful**	Specifies how the new firmware is to be activated. Choose between immediate or graceful reload.

Here's an example for configuration provisioning:

```
profile provisioning CONFIG
destination configuration
location 1 tftp://tftp1.provider.net/configs/$(system.mac).cfg location 2 tftp://172.16.1.33/configs/$(system.mac).cfg activa-
tion reload graceful
```

Explanation:

Step	Command	Purpose
1	[name] **(pf-prov)**[CONFIG]**#destination configuration**	Chooses the unit's startup-configuration as destination of the downloaded file.
2	[name] **(pf-prov)**[CONFIG]**#location 1 tftp://tftp1.provider.net /configs/ $(system.mac).cfg**	Specifies the location of the file to check for changes. $(system.mac) is a placeholder for the unit's MAC address of ETH 0/0. Using host names instead of IP addresses works only if DNS resolver is enabled and configured.

Step	Command	Purpose
3	[name] **(pf-prov)**[CONFIG]**#location 2 tftp://172.16.1.33/configs/$(system.mac).cfg**	Specifies alternate locations of the file. If the first could not be contacted, the second is tried, and so on.
4	[name] **(pf-prov)**[CONFIG]**#activation reload graceful**	Specifies how the new configuration should be activated. Choose between immediate or graceful reload.

Note the placeholder used in the file location. Placeholders can be used for each part of the location, be it server address, path or filename. The following place holders are available:

- **$(system.mac)**—MAC address of ETH 0/0 (without ":" between the hexadecimal characters)
- **$(system.serial)**—serial number of the unit
- **$(dhcp.66)**—DHCP option 66 (TFTP server IP), as delivered by the DHCP server (only if DHCP is enabled)
- **$(dhcp.67)**—DHCP option 67 (Boot file name), as delivered by the DHCP server (only if DHCP is enabled)

To use and debug provisioning:

Step	Command	Purpose
1	[name] **(cfg)provisioning execute FIRMWARE**	Executes the provisioning profile FIRMWARE once
2	[name] **(cfg)debug provisioning**	Enables debug output for all provisioning operations

To continuously poll for firmware or configuration changes, use the **provisioning execute** command together with the new **timer** command as described below. Here's how to do both firmware and configuration provisioning, with a polling interval of 10 minutes.

```
timer FIRMWARE_UPDATE now + 2 minutes every 10 minutes "provisioning execute FIRMWARE"
timer CONFIG_UPDATE now + 2 minutes every 10 minutes "provisioning execute CONFIG"
```

Boot procedure

During a normal boot procedure of an IPLink, the bootstrap application checks for an application image in the persistent memory of the logical region *nvram*:. The application image is then executed, i.e. the IPLink software is started module by module. One of the last start-up tasks to finish in bringing up the entire system is handling the operating configuration. The configuration *startup-config* is copied from the logical region *nvram:*

in nonvolatile memory to the logical region *running-config* in the volatile memory. The IPLink software now uses the *running-config* to set up the operating configuration of the IPLink. Figure 10 illustrates the boot procedure.

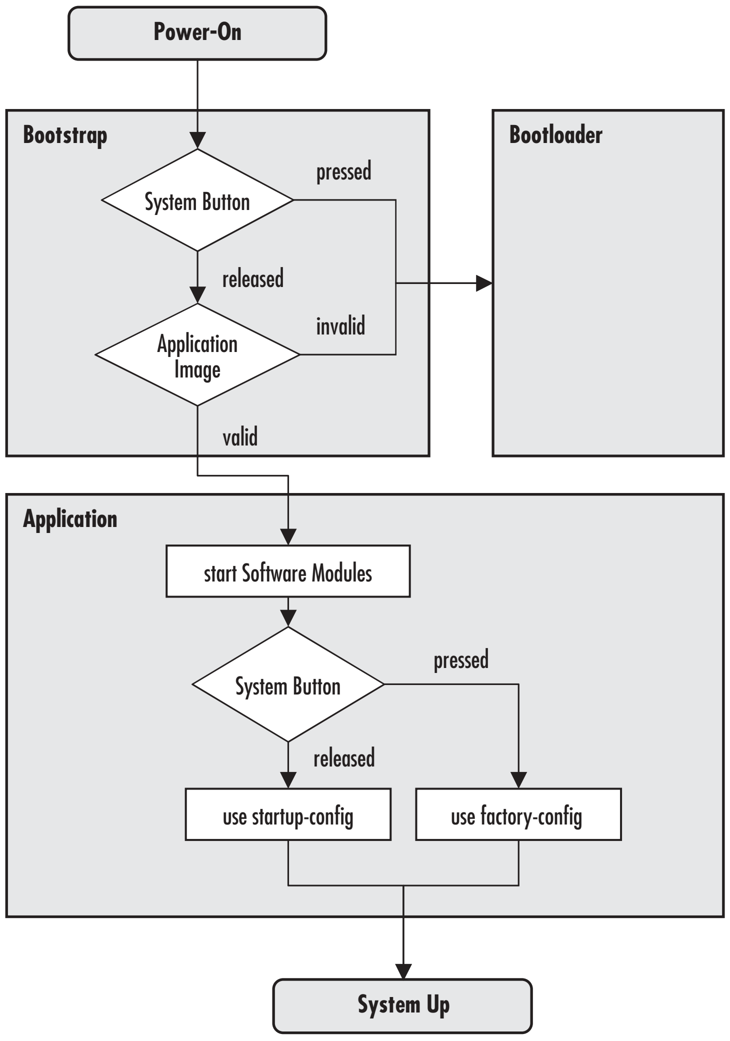


Figure 10. Boot procedure

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

- "If the user has pressed the system button, it launches the bootloader, the bootstrap application checks the status of the *Reset* button on the back panel of the IPLink."
- If a valid application image is not available

The bootloader ensures that basic operations, network access, and downloads are possible in case of interrupted or corrupted application image downloads.

After downloading an application image (that is, new system software/software upgrade), the bootloader ensures that basic operations, network access, and downloads are possible in case of interrupted or corrupted application image downloads. After downloading an application image, the bootstrap will only switch to the newly loaded application image if it is valid. If it is not valid, the bootstrap still uses the application image which existed prior to doing a software upgrade.

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

Bootloader

Recall that the bootloader ensures that basic operations, network access, and downloads are possible in case of interrupted or corrupted application image downloads. The IPLink Series comes with the *RedBoot Bootloader*. It offers new features such as console access to the Bootloader and the capability for downloading application images (e.g. IPLink software) via the serial link of the console.

Start Bootloader

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

```
RedBoot>
```

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

Note If the cursor keys (up, down, left, right) are not working, use **<ctrl>-<n>** (for up) and **<ctrl>-<p>** (for down) instead. Commands can be abbreviated as long as they do not become ambiguous.

Start-up with factory configuration

Step	Command	Purpose
1	**RedBoot> fis load**	Copies the IPLink software 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 IPLink software 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 (IPLink software) via TFTP

The following procedure downloads the application image (IPLink software) 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*	Tests the 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 IPLink could directly execute it. *host*: IP address of the TFTP server *base_address*: memory location where to store the application image. Use the 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	Deletes the first application image. Reply with 'y' to the confirmation request.
6	RedBoot> fis create	Stores the downloaded application image to the permanent memory (flash:). Reply with 'y' to the confirmation request.
7	RedBoot> fis list -l	Checks whether the image has been successfully stored, whether it is the desired Release and Build, and whether it is valid.

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

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

Note This procedure does not download the respective CLI description file. Download it after starting up IPLink software with the following command: `copy tftp://<tftp_server_address>/<server path>/b1 flash:`

Example: Downloading and storing a new application image (IPLink software)

```
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 /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          IPLink software R3.20
   0x01800100  0x01800100               V2.10

RedBoot> go
Starting 'IPLink software R3.20 at 0x01800100 via 0x01800100
```

Load a new application image (IPLink software) via the serial link

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

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

Factory configuration

IPLink devices 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 IPLink software, which makes sense at the very beginning. Moreover, in case of IPLink software malfunction, you can reset to the initial state by reloading the factory configuration. The factory configuration consists of the default settings for the IP networking subsystem.

Once the user-specific configuration is created and stored as startup configuration, the factory configuration is no longer used but it remains in the persistent memory. It is possible to switch back to the factory configuration at any time during the operation of an IPLink. See section "Boot procedure" on page 65 and section "Start-up with factory configuration" on page 68 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!

Chapter 7 Configuration file handling

Chapter contents

Introduction

This chapter describes how to upload and download configuration files from and to an IPLink device. A configuration file is a batch file of IPLink software commands used in the software modules that perform specific functions of the IPLink. This chapter also describes some aspects of configuration file management. Refer to chapter 6, "System image handling" on page 57 for more information.

This chapter includes the following sections:

- Factory configuration (see page 74)
- Configuration file handling task list (see page 74)

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

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

Changing the current running configuration is possible as follows:

- You may change the running configuration interactively. Interactive configuring requires that you access the CLI by using the **enable** command to enter administrator execution mode. You must then switch to the configuration mode with the command **configure**. Once in configuration mode, enter the configuration commands that are necessary to configure your IPLink.
- You can also create a new configuration file or modify an existing one offline. You can copy configuration files from the IPLink 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 IPLink network interfaces.

See chapter 4, "Accessing the CLI" on page 38 for information concerning access to the CLI.

The following sections focus on IPLink software memory regions and software components that can be copied within the memory or uploaded/downloaded between a TFTP server and the memory of the IPLink. Since IPLink software uses a specific vocabulary in naming those software components, refer to appendix A, "Terms and definitions" on page 304 to ensure that you understand the concepts. Refer to chapter 6, "System image handling" on page 57 for a brief description of how IPLink software uses system memory.

Understanding configuration files

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

Figure 11, shows the characteristics of a configuration file. It is stored on a TFTP server in the file *IP2805_001.cfg* for later download to the IPLink. The command syntax used to enter commands with the CLI and add commands in configuration files is identical. For better comprehension, you can add comments in configuration files. To add a line with a comment to your configuration file, simply begin the line with the hash (#) character. The command parser skips everything after the hash character to the end of the line.

```
#----------------------------------------------------------------#
#                                                                #
# 2805                                                           #
# Factory configuration file                                     #
#                                                                #
#----------------------------------------------------------------#

dns-relay
sntp-client
sntp-client server primary 129.132.2.21 port 123 version 4

profile napt NAPT

profile dhcp-server DHCP
  network 192.168.1.0 255.255.255.0
  include 1 192.168.1.10 192.168.1.99
  lease 2 hours
  default-router 1 192.168.1.1
  domain-name-server 1 192.168.1.1

context ip router

  interface eth0
    ipaddress dhcp
    use profile napt NAPT
    tcp adjust-mss rx mtu
    tcp adjust-mss tx mtu

  interface eth1
    ipaddress 192.168.1.1 255.255.255.0
    tcp adjust-mss rx mtu
    tcp adjust-mss tx mtu

context ip router
  dhcp-server use DHCP

port ethernet 0 0
  medium auto
  encapsulation ip
  bind interface eth0 router
  no shutdown

port ethernet 0 1
  medium auto
  encapsulation ip
  bind interface eth1 router
  no shutdown
```

Figure 11. Sample configuration file

Each configuration file stored in the flash memory needs a unique name. The user has to assign a file name to any user-specific configuration. IPLink software predefines some names for configuration files. These are the factory configuration (*factory-config*), startup configuration (*startup-config*), and running configuration (*running-config*) file names. Refer to appendix A, "Terms and definitions" on page 304 to learn more about configuration file types.

Factory configuration

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

- Default settings for the IP networking subsystem
- Default settings for the quality of service subsystem

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

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 IPLink 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 IPLink device to suit your own operating requirements.

The tasks in this chapter assume that you have at least a minimal configuration running on your system. You can create a basic configuration file by using the **configure** command; see section "Modifying the running configuration at the CLI" on page 80 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 75)
- Replacing the startup configuration with a configuration from the Flash memory (see page 76)
- Copying configurations to and from a remote storing location (see page 78)
- Replacing the startup configuration with a configuration downloaded from the TFTP server (see page 79)
- Displaying configuration file information (see page 80)
- Modifying the running configuration at the CLI (see page 80)
- Modifying the running configuration offline (see page 81)
- Deleting a specified configuration (see page 83)

- Downloading encrypted files (see page 83)

Copying configurations within the local memory

Configuration files may be copied into the local memory in order to switch between different configurations. Remember the different local memory regions in IPLink software as shown in figure 12.

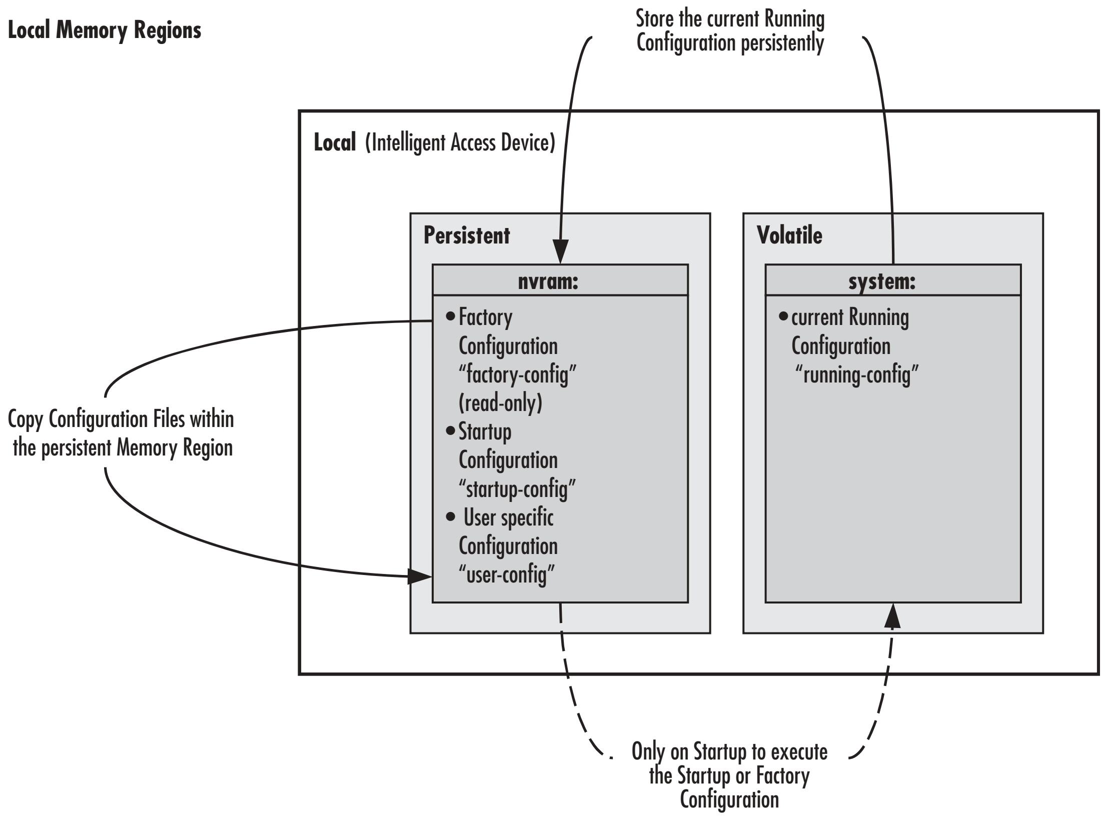

Local Memory Regions
Store the current Running Configuration persistently
Local (Intelligent Access Device)
Persistent
nvram:
• Factory Configuration "factory-config" (read-only)
• Startup Configuration "startup-config"
• User specific Configuration "user-config"
Volatile
system:
• current Running Configuration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

Figure 12. Local memory regions in IPLink software

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

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

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

As shown in figure 12 the local memory regions are identified by their unique names, like *nvram:,* which is located in flash memory, and *system*:, which is the system RAM, i.e. the volatile memory. As already mentioned, configuration files in the same memory region need a unique name. For example, it is not possible to have two configuration files with the name *running-config* in the memory region *nvram:*.

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

Mode: Administrator execution

Step	Command	Purpose
1	***node#*copy {factory-config | startup-config | running-config | nvram:** *source-name* **} nvram:***target-name*	Copies the selected source configuration file *source-name* as target configuration file *target-name* into the local memory.

Example: Backing up the startup configuration

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

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

Replacing the startup configuration with a configuration from Flash memory

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

Mode: Administrator execution

Step	Command	Purpose
1	***node#* copy nvram:***new-startup* **startup-config**	Replaces the existing persistent startup configuration with the startup configuration *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 by using the **copy** command.

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

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

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

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

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

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

  cli version 3.00
  snmp community public rw
…
framerelay
  exit

IPLink#
```

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 IPLink and the remote data store on a server system (see figure 13). A remote storage location is mostly used to store ready configurations for later download to a certain IPLink. A TFTP server has to be used as a remote data store. From within IPLink software, 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.

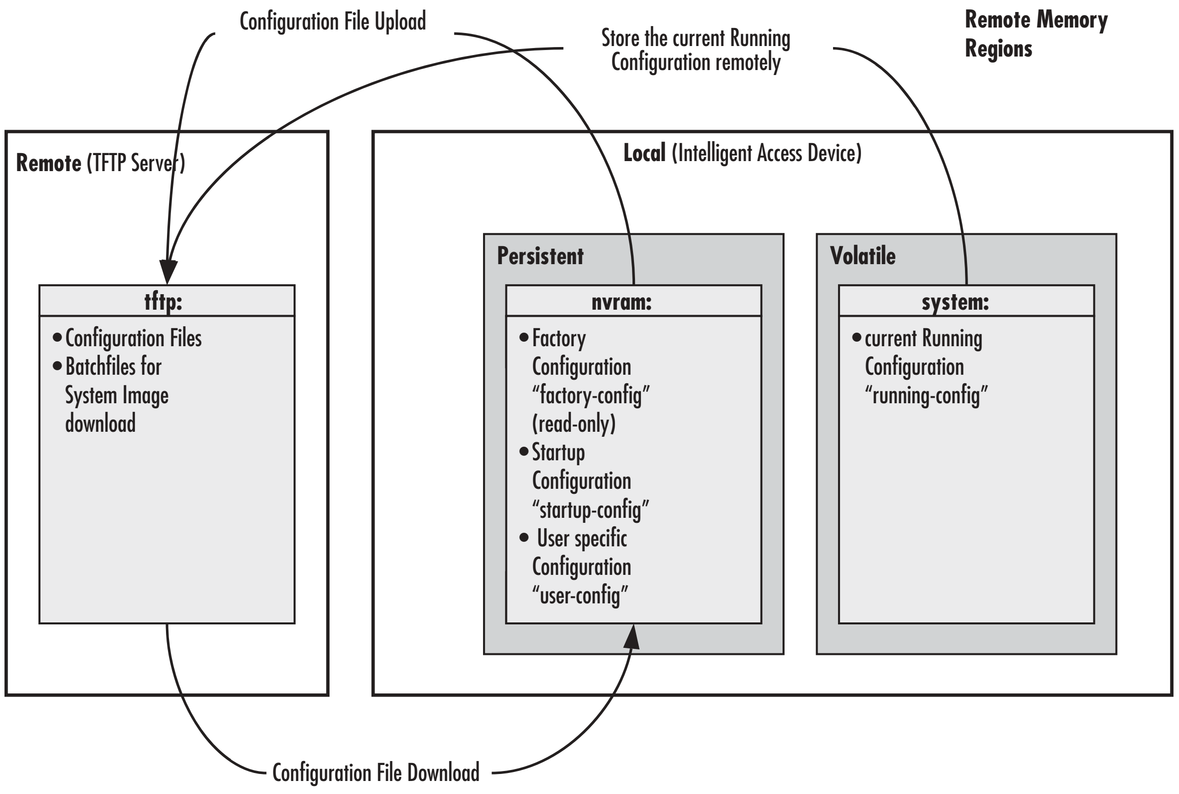

Configuration File Upload
Store the current Running Configuration remotely
Remote Memory Regions
Remote (TFTP Server)
Local (Intelligent Access Device)
Persistent
Volatile
tftp:
• Configuration Files
• Batchfiles for System Image download
nvram:
• Factory Configuration "factory-config" (read-only)
•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

Figure 13. Remote memory regions for IPLink software

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 IPLink 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 IPLink according to your requirements, the running configuration of this device, named *running-config* and located in the volatile memory region *system:,* is edited. Next, the configuration is tested and if everything is as required, the running configuration is copied as startup configuration, named *startup-config*, into the persistent memory region *nvram:* of the target device. After this, the startup configuration is transferred to the

TFTP server, where it can be distributed to other IPLink devices. These devices therefore get clones of the starting system if the configuration does not need any modifications.

Replacing the startup configuration with a configuration downloaded from TFTP server

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

Mode: Administrator execution

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

Example: Sample configuration download from the TFTP server

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

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

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

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

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

  cli version 3.00
  snmp community public rw
…
…
framerelay
  exit

IPLink#
```

Displaying configuration file information

This procedure describes how to display information about configuration files

Mode: Administrator execution

Command	Purpose
show nvram:	Lists all persistent configurations
show running-config	Displays the contents of the running configuration file
show startup-config	Displays the contents of the startup configuration file

IMPORTANT

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

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

Modifying the running configuration at the CLI

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

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

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

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

Mode: Administrator execution

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

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

Modifying the running configuration offline

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

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

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

Mode: Administrator execution

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

Example: Modifying the running configuration offline

The following example shows how to upload the running configuration from the IPLink to the file *current-config* on a TFTP server at IP address 172.16.36.80. The uploaded configuration file is written into the root directory specified by the TFTP server settings, and overwrites any existing file with the same name. Read your TFTP server manual to get a thorough understanding of its behavior. After this, the configuration file is available for offline editing on the TFTP server. Once the configuration file *current-config* has been modified, it is downloaded from the TFTP server, at IP address 172.16.36.80, into the IPLink's persistent memory region *nvram:* using the name *startup-config*. Finally, you must restart the IPLink.

```
IPLink#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.

```
IPLink#copy tftp://172.16.36.80/user/ current-config nvram:startup-config
Download...100%
IPLink#reload
Press 'yes' to restart, 'no' to cancel : yes
```

```
The system is going down
```

Deleting a specified configuration

This procedure describes how to delete configuration files from the IPLink 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 the flash memory.

Example: Deleting a specified configuration

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

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

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

2. Delete the configuration named *minimal* explicitly.

```
IPLink#erase nvram:minimal
```

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

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

Encrypted file download

This section explains the encrypted configuration download feature of IPLink software.

TFTP as a configuration download mechanism has the advantage of being extremely simple (trivial) and applicable in any network without any requirements for specialized management servers or applications. It has the disadvantage of being completely insecure.

The security hole of downloading complete configurations—which may contain IP addresses, login names and passwords for PPP or VoIP registrations—using TFTP becomes particularly pressing in combination with the auto-provisioning feature which allows large scale distribution of configurations in entire networks.

To alleviate this problem and maintain the simplicity of TFTP downloads support for encrypted configuration file downloads is introduced.

Goal: Prevent maliciously intercepted configurations to be readable by unauthorized users.

Pre-requisites: Only authorized users have configuration access to the IPLink. The configurations can be stored in plain form on the IPLink. SNMP Write Access shall be restricted by means of communities and ACLs to prevent unauthorized SNMP initiated configuration downloads. Telnet access shall be restricted by means of credentials and ACLs.

Encrypted Configuration Download

An external encryption tool on the PC is used to encrypt the configuration file:

```
enctool encrypt <plain-config-file> <enc-config-file> [<key>]
```

The encrypted configuration file can then be downloaded with TFTP triggered by

- The CLI copy command: **copy tftp://<host>/<path> <config-file>**
- Auto provisioning
- SNMP
- HTTP

On the IPLink the encryption is detected and the configuration file is automatically decrypted before stored to flash.

A custom encryption key can be:

- Downloaded to the IPLink software
- Specified with the PC encryption tool

The encryption key may include the MAC address and/or serial number of the IPLink using the placeholders $(system.mac) and $(system.serial) respectively.

An encrypted configuration file can be uploaded to a TFTP server on request, specifying the encrypted flag:

```
copy <config-file> tftp://<host>/<path> encrypted
```

On the PC the encryption tool can be used to decrypt the file:

```
enctool decrypt <enc-config-file> <plain-config-file> [<key>]
```

A log file lists the last up/downloads:

```
show log file-transfer
```

Use Cases

Install a custom encryption key (optional)

You can install a custom encryption key with the IPLink. The encryption key is used to automatically decrypt an encrypted configuration file that is downloaded later. A default encryption key is already installed on the IPLink.

To install an encryption key you have to create a file on your TFTP server that contains the key. Then you have to download this key file to the IPLink using the **copy** command of the IPLink.

The key file shall contain a key string of at most 24 characters on a single line. Spaces, tabs and LF/CR characters are trimmed. The key must not contain LF/CR or the null character and must not start or end with a space or tab. If the key contains more than 24 characters, only the first 24 characters are considered.

The key may contain variables that are resolved when the key file is downloaded to an IPLink. Using this mechanism you can specify device-specific encryption keys. We currently support the following variables:

- **$(system.mac)**: The MAC address of the first ethernet port. Execute the **show port ethernet** command on an IPLink to display the MAC address of an IPLink. This value without the colon separators and with all lower-case hexadecimal letters is used instead of the variable on the IPLink.
- **$(system.serial)**: The serial number of the IPLink. Execute the **show version** command on the IPLink to display the serial number.

When your key file contains the following line:

```
123$(system.serial)abc$(system.mac)XYZ
```

The command **show port ethernet** shows the following:

```
Ethernet Configuration
-------------------------------------
Port           : ethernet 0 0 0
State          : OPENED
MAC Address    : 00:0C:F1:87:D9:09
Speed          : 10MBit/s
Duplex         : Half
Encapsulation  : ip
Binding        : interface eth0 router
```

The command **show version** displays the following:

```
Productname      : SN1200
Software Version : R3.20 TB2005-06-24_MEYER SIP
Supplier         :
Provider         :
Subscriber       :

Information for Slot 0:
SN1200
Hardware Version : 0004, 0001
Serial number    : 100000020002
Software Version : R3.20 TB2005-06-24_MEYER SIP
```

The encryption key on this IPLink will be interpreted as:

```
123100000020002abc000cf187d909XYZ
```

Then you have to download the created key file to the IPLink. Open a telnet session and type in the following commands:

```
>enable
#copy tftp://<ip>/<path> key:
```

where *<ip>* is the IP address of your TFTP server and *<path>* is the path to the key file relative to the TFTP root.

IMPORTANT

The downloaded key also defines how the passwords are encrypted in your configuration files. After you downloaded a key file you have to regenerate the **startup-config** from the **running-config** by executing the command.

```
copy running-config startup-config
```

If you don't do this, the device will fail executing the commands that have encrypted password arguments, e.g., 'administrator', 'h235-security password', etc.

Encrypt a configuration file

Use the encryption tool to encrypt a configuration file on your PC. Therefore you have to enter the following command.

```
enctool encrypt <plain-file> <encrypted-file> [<key>]
```

Where <plain-file> is the path of the non-encrypted input configuration file and <encrypted-file> is the path of the encrypted output configuration file. <key> specifies the encryption key which shall be used to encrypt the configuration file. If omitted the default key is used.

Download an encrypted configuration file

Now you can download the configuration file as usual using the CLI copy-command, the auto-provisioning feature, HTTP or SNMP download. The IPLink automatically detects that a downloaded file is encrypted and tries to decrypt the file using the pre-installed key.

Upload an encrypted configuration file

The IPLink immediately decrypts a configuration file after downloading it. This is the configuration file is stored non-encrypted in the flash memory. Thus when you upload a configuration it is uploaded non-encrypted.

You may upload an encrypted configuration file specifying the encrypted flag at the end of the copy command:

```
#copy startup-config tftp://<ip>/<path> encrypted
```

This encrypts the configuration file before sending it to the TFTP server. Use the **enctool decrypt** command on the PC to regain the original configuration.

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. The following are basic IPLink software parameters that must be established when setting up a new system:

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

Additionally, the following tasks are described in this chapter:

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

Basic system management configuration task list

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

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

- Managing feature license keys (see page 89)
- Setting system information (see page 90)
- Setting the system banner (see page 91)
- Setting time and date (see page 92)
- Displaying clock information (see page 92)
- Displaying time since last restart (see page 93)
- Configuring and starting the web server (see page 93)
- Determining and defining the active CLI version (see page 93)
- Restarting the system (see page 94)
- Displaying the system event log (see page 95)
- Controlling command execution (see page 95)
- Setting timed execution of CLI commands (see page 97)
- Displaying the checksum of a configuration (see page 97)
- Configuration of terminal sessions (see page 97)

Managing feature license keys

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

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

Mode: Configure

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

Example: Installing license keys from a TFTP server

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

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

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#install license** *license-key*	Install the license key
2		Repeat step 1 for any additional license keys

Example: Installing license keys from the console

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

```
IPLink(cfg)#install license 10011002R1Ws63yKV5v28eVmhDsVGj/JwKqIdpC4Wr1BHaNtenXUYF/
2gNLoihifacaTPLKcV+uQDG8LJis6EdW6uNk/GxVObDEwPFJ5bTV3bIIfUZ1eUe+8c5OpCCd7PSAe83Ty2c/
CnZPSlEjIrVlJrr8VhOr1DYxkEV9evBp+tSY+y9sCeXhDWt5Xq15SAPlznTLQmym7fDakvm+zltzswX/
KX13sdkR0ub9IX4Sjn6YrvkyrJ2dCGivTTB3iOBmRjV1u
```

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

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#show licenses**	Display all installed licenses

Example: Displaying installed licenses

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

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

Setting system information

The system information includes the following parameters:

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

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

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

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

Assigning explanatory location information to describe the system physical location of your IPLink (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 IPLink device, together with information on how to contact this provider. The provider is a company making services available to subscribers. The provider information may be any alphanumeric string, including spaces, that is no longer than one line. This entry corresponds to the Patton Electronics enterprise-specific MIB provider object.

The system subscriber information is used to get in touch with subscriber for this IPLink device, together with information on how to contact this subscriber. The subscriber is a company or person using one or more services from a provider. The subscriber information may be any alphanumeric string, including spaces, that is no longer than one line. This entry corresponds to the Patton Electronics enterprise-specific MIB subscriber object.

The system supplier information is used to get in touch with the supplier for this IPLink device, together with information on how to contact this supplier. The supplier is a company delivering IPLink devices to a provider. The supplier information may be any alphanumeric string, including spaces, that is no longer than one line. This entry corresponds to the Patton Electronics enterprise-specific MIB supplier object.

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#system contact** *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 the system information must have more than one word, enclose it in double quotes.

Example: Setting system information

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

```
IPLink(cfg)#system contact "Bill Anybody, Phone 818 700 1504"
IPLink(cfg)#system hostname IPLink
IPLink(cfg)#system location “Wiring Closet, 3rd Floor”
IPLink(cfg)#system provider “Best Internet Services, contact@bis.com, Phone 818 700 2340”
IPLink(cfg)# system subscriber “Mechanical Tools Inc., jsmith@mechtool.com, Phone 818 700
1402”
IPLink(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 IPLink via Telnet or a serial connection (see figure 14). 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 14. 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.

```
IPLink(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 IPLink 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 IPLink software includes an integrated SNTP client, which allows synchronization of time-of-day and date to a reference time server. Refer to chapter 21, "SNTP client configuration" on page 240 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.

```
IPLink(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.

```
IPLink>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.

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

Configuring and starting the Web server

IPLink includes an embedded web server, that can be used together with a customer-specific Java applet that must be downloaded into the persistent memory region of your IPLink. 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 IPLink software are possible using a browser instead of accessing the IPLink software 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 IPLink software.

Mode: Configure

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

Example: Configuring and starting the Web server

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

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

Determining and defining the active CLI version

IPLink software 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.

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

Restarting the system

In case the IPLink 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 IPLink that is to be restarted.

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

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

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.

```
IPLink#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 IPLink software. 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 IPLink.
5	***node*#show log login**	Displays a list of succeeded and failed CLI login attempts.
6	***node*#show log file-transfer**	Displays the history of all recently executed file transfer operations (up to 50 entries).

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.

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

Controlling command execution

The IPLink software command shell includes a basic set of commands that allow you to control the execution of other running commands. In IPLink software, 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

Step	Command	Purpose
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.

```
IPLink>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

```
IPLink>show ip interface
------------------------------------------------------------
Context:                 router
…
```

Show the currently running commands

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

Bring job 0 to foreground

```
IPLink>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)
```

Timed execution of CLI command

The command **timer** allows the timed execution of CLI commands. The **timer** command is incremental; this means for each time it is entered, a new timer is created. All timers appear in the running-configuration, except if they have been created with the *volatile* option. It is possible to specify for each timer the start time and the reoccurrence. Use the CLI help (tab completion) for detailed description of all configuration options.
Some examples:

```
timer FIRMWARE_UPDATE now + 2 minutes every 10 minutes “provisioning execute FIRMWARE”
```

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

```
timer volatile RELOAD midnight + 1 hour “reload graceful”
```

Starts a volatile timer named *RELOAD* (does not appear in the running-configuration, and thus is not stored in the startup-configuration). The timer is executed once, 1 hour after midnight, and reloads the system gracefully.

Displaying the checksum of a configuration

In IPLink software 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.

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

Configuration of terminal sessions

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

Mode: System

Step	Command	Purpose
1	[*name*] **(sys)#terminal height**	Configures the terminal height.

Step	Command	Purpose
2	[*name*] **(sys)#[no] terminal idle-time-logout**	After 30 minutes without user input, a terminal session is automatically closed. If longer session periods are required (logging/debugging) this command allows to increase the session timeout, or to disable it completely.
3	[*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 RADIUS Client Configuration

Chapter contents

Introduction

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

To use the afsuthentication and authorization service on IPLink software you have to configure the AAA component, the RADIUS component and the local database accounts.

This chapter includes the following sections:

- The AAA component
- RADIUS configuration (see page 103)
- Configuration of the local database accounts (see page 108)

The AAA component

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

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

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

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

AAA information can be stored in a local database or in a database on a remote server. A current standard by which network access servers interface with the AAA server is the Remote Authentication Dial-In User Service (RADIUS). IPLink software supports local database and RADIUS AAA.

Currently, the IPLink software AAA component is used only by the login service. Authentication verifies the user by password, authorization grants access to the command line interface at administrator or operator levels. The IPLink gets the AAA information from the local database or from one ore more RADIUS servers.

Figure 15 illustrates the authentication procedure for a user logging into an IPLink that is configured to use RADIUS as authentication method.

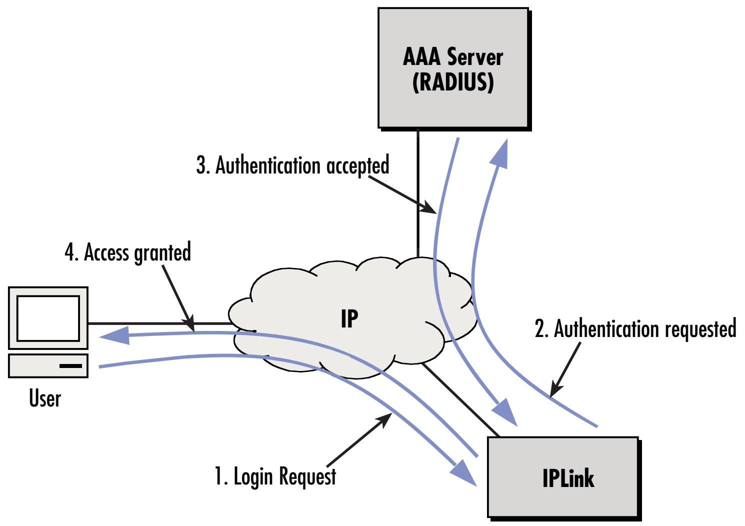


Figure 15. Authentication procedure with a RADIUS server

General AAA Configuration

The AAA component consists of *AAA profiles* and *AAA methods*. A service (e.g. Telnet) has to specify a profile it wants to apply to all login requests. The profile then specifies the sequence in which methods are applied to obtain AAA information. Figure 16 illustrates the correlation between the Telnet login and console login services.

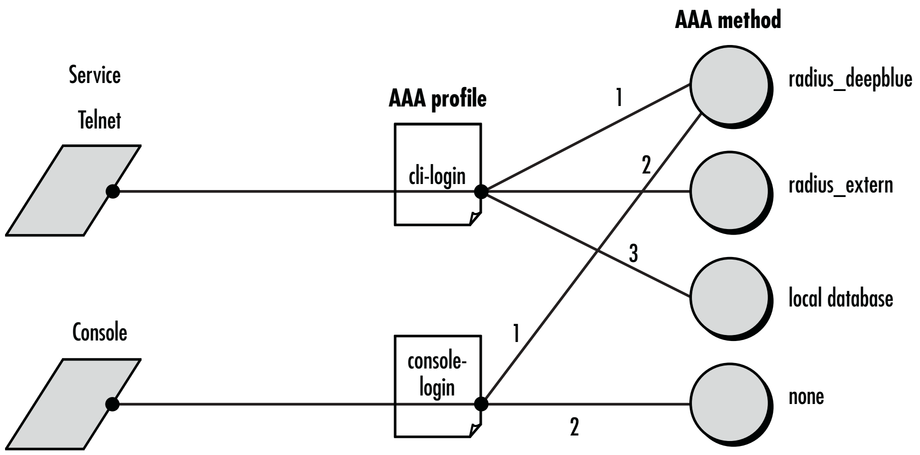


Figure 16. How to use AAA methods and AAA profiles

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

1. Query RADIUS server *radius_deepblue*.
2. Query RADIUS server *radius_extern*.

3. Query the local database (see “Configuring the local database accounts” on page 108 for information on how to configure the local database)

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

1. Ask radius server *radius_deepblue*.
2. Ask predefined method *none*. This method always grants access as system operator.

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

Do the following to configure the AAA component.

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#profile aaa** *name*	Creates an authentication profile with name *name* and enters profile authentication configuration mode.
2	**node(pf-auth)[name]#method** [*index*] **{local | none | {radius** *name***}}**	Adds an AAA method to the profile. For RADIUS you have to specify a name. For information on how to configure local accounts and RADIUS servers, refer to chapter 10, “IP context overview” on page 110. With *index* you can add a method between to others.
3		Repeat step 2 for all AAA methods you want to add
4	**node(pf-auth)[name]#server-timeout** *seconds*	Sets the timeout after that the next AAA method in the list is requested if no answer is received.
5	**node(pf-auth)[name]#exit**	Goes back to the parent configuration mode
6	***node*(cfg)#terminal Telnet use aaa** *profile-name*	Specifies which AAA profile the Telnet login service has to use.
7	***node*(cfg)#terminal console use aaa** *profile-name*	Specifies which AAA profile the console login service has to use.
8	***node*(cfg)#show profile aaa** [*name*]	Displays the configured profiles

Example: Create the AAA profiles for login over Telnet and login over console, as they are shown in figure 16, and use them on the Telnet login and console login services.

```
IPLink>enable
IPLink#configure
IPLink(cfg)#profile aaa remote-radius
IPLink(pf-aaa)[remote-~]#method radius radius_deepblue
IPLink(pf-aaa)[remote-~]#method radius radius_extern
IPLink(pf-aaa)[remote-~]#method local
IPLink(pf-aaa)[remote-~]#server-timeout 15
IPLink(pf-aaa)[remote-~]#exit
IPLink(cfg)#
IPLink(cfg)#profile aaa local-only
IPLink(pf-aaa)[local-o~]#method local
IPLink(pf-aaa)[local-o~]#method none
IPLink(pf-aaa)[local-o~]#exit
IPLink(cfg)#terminal Telnet use aaa remote-radius
IPLink(cfg)#terminal console use aaa local-only
IPLink(cfg)#show profile aaa

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

Authentication Profile: remote-radius
  Server-Timeout: 15
  Methods:
    radius_deepblue (Type=radius)
    radius_extern (Type=radius)
    local (Type=local)

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

IPLink(cfg)#
```

IMPORTANT

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

Note If you do not configure AAA, a default AAA profile exists containing the *AAA local* as the first AAA method and the *AAA none* as the second. The Telnet login and the console login service use this profile. If an emergency occurs, you can reload this default configuration by reloading the factory configuration as described in section "Boot procedure" on page 65.

RADIUS configuration

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

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

Configuring RADIUS clients

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

Mode: Configure

Step	Command	Purpose
1	***node***(**cfg)#radius-client** *name*	Adds a RADIUS client with name *name* and enters RADIUS-client configuration mode
2	**node(radius)[name]#radius-server** *hostname*	Sets the hostname (or IP address) of the remote RADIUS server
3	**node(radius)[name]#shared-secret authentication** *secret*	Sets the password shared between the RADIUS client (the IPLink) and the remote RADIUS server.
4	**node(radius)[name]#exit**	Goes back to the parent configuration mode
5	**node(cfg)#show radius-client** *name*	Displays configured RADIUS servers

Example: Configure the RADIUS clients as shown in figure 16.

```
IPLink>enable
IPLink#configure
IPLink(cfg)#radius-client radius_deepblue
IPLink(radius)[radius_~]#radius-server deepblue
IPLink(radius)[radius_~]#shared-secret authentication 78f8a23b
IPLink(radius)[radius_~]#exit
IPLink(cfg)#radius-client radius_extern
IPLink(radius)[radius_~]#radius-server 219.144.12.1
IPLink(radius)[radius_~]#shared-secret authentication dd9351e13cc335
IPLink(radius)[radius_~]#exit
IPLink(cfg)#
IPLink(cfg)#show radius-client
RADIUS clients:
  radius_deepblue
  radius_extern
IPLink(cfg)#show radius-client radius_deepblue
AAA RADIUS Module: radius_deepblue
  Authentication Shared Secret: 78f8a23b
  Timeout: 6+
  Sessions:
  UDP Interface:
    Configured Server Hostname: deepblue
IPLink(cfg)#show radius-client radius_extern
AAA radius Module: radius_extern
  Authentication Shared Secret: dd9351e13cc335
  Timeout: 6
  Sessions:
  UDP Interface:
    Configured Server Hostname: 219.144.12.1
```

```
IPLink(cfg)#
```

Configuring RADIUS accounting

The RADIUS accounting functionality can be added to a call-router configuration by inserting an AAA call-control service between two call-router elements. Any call that is then routed through the AAA service will cause call detail records (CDRs) to be sent to the radius server. Normally an accounting start record is sent when the call is connected and the accounting stop record is sent, when the call is disconnected. If enabled, the AAA service is also able to send interim update records, after a specified interval. The AAA service can include the following standard RADIUS attributes in the CDRs:

```
ATTRIBUTE Acct-Status-Type
ATTRIBUTE Acct-Session-Time
ATTRIBUTE Acct-Session-Id
ATTRIBUTE NAS-Identifier
ATTRIBUTE Called-Station-Id
ATTRIBUTE Calling-Station-Id
```

Additionally, the following vendor specific attributes are available to support voice service specific information:

```
#
# dictionary.patton
#
VENDOR        Patton           1768
#
#             Name             Id       Type     Vendor Note
#
ATTRIBUTE Setup-Time 32 string Patton a)
ATTRIBUTE Connect-Time 33 string Patton a)
ATTRIBUTE Disconnect-Time 34 string Patton a)
ATTRIBUTE Disconnect-Cause 35 integer Patton b)
ATTRIBUTE Disconnect-Source 36 string Patton c)
ATTRIBUTE Called-Unique-Id 48 string Patton d)
ATTRIBUTE Called-IP-Address 49 ipaddr Patton
ATTRIBUTE Called-Numbering-Plan 50 string Patton e)
ATTRIBUTE Called-Type-Of-Number 51 string Patton f)
ATTRIBUTE Calling-Unique-Id 80 string Patton d)
ATTRIBUTE Calling-IP-Address 81 ipaddr Patton
ATTRIBUTE Calling-Numbering-Plan 82 string Patton e)
ATTRIBUTE Calling-Type-Of-Number 83 string Patton f)
ATTRIBUTE Calling-Presentation-Indicator 88 string Patton g)
ATTRIBUTE Calling-Screening-Indicator 89 string Patton h)

a) Format of timestamps is "WWW MMM DD HH:MM:SS YYYY" Example: "Wed Jun 15 09:20:55 2005"
b) ITU-T Q.931 cause value (1-127)
c) { originator | terminator | internal }
d) Contains the Call-Id for SIP or H.323
e) { e.164 | data | telex | national | private }
f) { international | national | network specific | subscriber | abbreviated }
g) { allowed | restricted | unavailable }
h) { user-provided, not screened | user-provided, verified and passed | user-provided, verified and failed | network provided }
```

Note The subset of information elements that is actually included in a CDR is dependant on the type of call and the information already available at the time the CDR is sent.

The following procedure guides you through the steps necessary to enable RADIUS accounting in an existing configuration:

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)# radius-client <client-name>**	Create a new RADIUS client
2	***node*(radius)[client-*name*]# radius-server <server-name-or-ip> [<udp-port>]**	Define the RADIUS server to be used. If the UDP port is omitted, the default port 1812 is used. **Note** For accounting RADIUS servers often use port 1813) **Note** There might also be RADIUS servers, which still use the old RADIUS ports 1645 or 1646)
3	***node*(radius)[client-*name*]# shared-secret authentication <secret>**	Define the shared secret to access the RADIUS server
4	***node*(radius)[client-*name*]# profile aaa <pf-name>**	Create an AAA profile, which uses the RADIUS client
5	***node*(pf-auth)[pf-*name*]# method radius <radius-client-name>**	Define your newly created radius client as the AAA method to be used. **Note** If you require redundancy, you can create multiple radius clients and add all of them to the AAA profile.
6	***node*(pf-auth)[pf-*name*]# context cs**	Switch to the circuit-switching context.
7	***node*(ctx-cs)[ctx-name]# service aaa <name>**	Create an AAA call-control service
8	***node*(svc-aaa)[svc-name]# accounting use profile <aaa-profile-name>**	Define the newly created AAA profile to be used for accounting using this AAA service.
9	***node*(svc-aaa)[svc-name]# nas-identifier <nas-identifier>**	Define the NAS-Identifier string to be included in RADIUS requests sent from this AAA service.
10 (Optional)	***node*(svc-aaa)[svc-name]# authentication use profile <aaa-profile-name>**	Optionally, you can also configure the AAA service to request authentication using the calls calling E.164 number. If this is required, you can define the AAA profile used for authentication using this command.
11 (Optional)	***node*(svc-aaa)[svc-name]# accounting-failure-action [drop-calls | ignore]**	Define, if calls shall be dropped, if accounting fails. The default is to ignore accounting failures.

Step	Command	Purpose
12 (Optional)	***node***(svc-aaa)[svc-name]# **accounting-start-trigger [setup | connect]**	Define, if accounting shall be started at call-setup or call-connect time. The default is at call-connect time. **Note** If setup is specified, an interim update will be sent at call-connect time. **Note** The Acct-Session-Time is always calculated from call-connect to call-disconnect time)
13 (Optional)	***node***(svc-aaa)[svc-name]# **[no] interim-update-interval <seconds>**	Define the interval, after which an interim update shall be sent, if necessary. The default is not to send periodic interim updates.
14	***node***(svc-aaa)[svc-name]# **port <name>**	Create a port for the routing path, you want to route through the AAA service.
15	**node(port)[port-name]# route call-dest- .....**	Define the routing destination for all calls received over this port.
16	**node(svc-aaa)[svc-name]# accounting-start-trigger [setup | connect]**	Go to the routing element, which is the source of the traffic to be sent to this AAA service and configure its routing destination to this AAA service port using the following command: **route call dest-service <service-name>.<port-name>**
17		Repeat steps 14 to 16 for each for each additional routing path you want to route through the AAA service

Configuring the RADIUS server

Each message to and from a RADIUS server includes several attributes. Attributes are, For example, in a login request, the name and password of the user that requires to log in. Each attribute is assigned a number by RFC 2865. This section gives an overview of all such attributes that IPLink software uses. For more information about each attribute, or other possible attributes, see RFC 2865 or the documentation of the radius server you use.

Attributes in the RADIUS request message

The IPLink sends a RADIUS request with the following attributes:

Attribute number	Attribute Type	Description
1	User-Name	Indicates the name of the user to be authenticated
2	User-Password	Indicates the password of the user to be authenticated
26	Protocol	Is a vendor specific attribute that indicates the protocol with that the user wants to log on. Currently it can have the value 'console' or 'Telnet'. Thus it is possible for the RADIUS Server to grant access depending on whether the user wants to log on over console or Telnet

Attributes in the RADIUS accept message

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

Attribute number	Attribute Type	Description
6	Service-Type	If the value is set to 'administrative', the user has administrator rights on the IPLink, otherwise operator rights
18	Reply-Message	Contains the text that is printed to the user after login. If the attribute is not included in the message, no text will be printed
27	Session-Timeout	Number of seconds the user is allowed to logged on. If the attribute is not included, the default value is infinite
28	Idle-Timeout	Number of seconds to stay in idle state before automatic logout proceeds. If the attribute is not included, the default value is 30 minutes. The command **terminal idle-time-logout** overwrites the value set by the attribute

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

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |  Length       |            Vendor-Id
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     Vendor-Id (cont)           |  Vendor-Type  | Vendor-Length |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Vendor-String ...
+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Type: 26

Length: Length of the whole attribute including the vendor data

Vendor-Id: 1768

Vendor-Type: 16

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

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

Configuring the local database accounts

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

Mode: Configure

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

Example: Create an administrator and an operator account

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

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

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

Chapter 10 IP context overview

Chapter contents

Introduction

This chapter outlines the IPLink software *Internet protocol (IP) context* and its related components. You will get the fundamental understanding on how to set up your IPLink to make use of IP related services.

The following sections describe the configuration steps necessary to put together certain IP services and the references to the related chapters that explain the issue in more details.

To understand the information given in the following chapters, carefully read to the end of the current chapter. Before proceeding, make sure that you feel comfortable with the underlying IPLink software configuration concept by reading chapter 2, "Configuration concepts" on page 29.

The IP context in IPLink software is a high level conceptual entity that is responsible for all IP-related protocols and services for data and voice. The IP context performs much the same function as a standalone IP router, and since every context is defined by a name, the IP context is named *router* by default. This IP context can contain interface static routes, RIP parameters, NAPT, QoS and access control profiles.

In figure 17 on page 111, the IP context with all its related elements is contained within the grayed area.

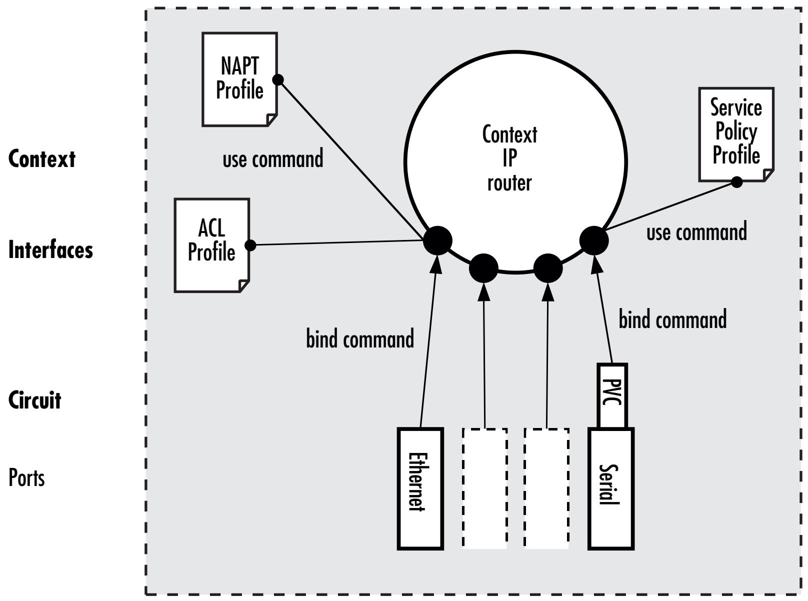


Figure 17. IP context and related elements

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

IP context overview configuration task list

As previously described, this chapter outlines the IP context configuration. It does not give you all the details of a configuration task, but refers you to the chapters in which you will find the full description.

- You can find all the information you need to configure an IP Interface in chapter 11, "IP interface configuration" on page 117.

- You can find the information regarding network address port translation (NAPT) in chapter 12, "NAT/ NAPT configuration" on page 128.
- If you need to configure a physical port, chapter 13, "Ethernet port configuration" on page 137 or chapter 15, "Serial port configuration" on page 167 may prove helpful.
- To set up the IP router contained in IPLink software, chapter 17, "Basic IP routing configuration" on page 195 and chapter 18, "RIP configuration" on page 201 give you the required information.
- For essential knowledge related to network security requirements, refer to chapter 19, "Access control list configuration" on page 211.
- If your network shall provide better service to selected network traffic, chapter 14, "Link scheduler configuration" on page 148 will help you to get in-depth knowledge about quality of service (QoS) management with IPLink software.

The following sections describe the basic tasks involved in IP context configuration. Many parameters have acceptable default values, which in most cases do not need to be explicitly configured. Hence not all of the configuration tasks below are required. Depending on your application scenario, some tasks are mandatory or might be optional. The following tasks use a bottom-up approach, starting from the ports, followed by the interfaces up to the services running on the IPLink. 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 112)
- Configuring Ethernet and serial ports (see page 113)
- Creating and configuring IP interfaces (see page 113)
- Configuring NAPT (see page 114)
- Configuring static IP routing (see page 114)
- Configuring RIP (see page 114)
- Configuring access control lists (see page 115)
- Configuring quality of service (see page 115)

Planning your IP configuration

The following subsections provide network connection considerations for several types of physical ports types. Patton recommends that you draw a network overview diagram displaying all neighboring IP nodes and serial connected elements. Do not begin configuring the IP context until you have completed the planning of your IP environment.

IP interface related information

Setting up the basic IP connectivity for your IPLink 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 address of the central TFTP server used for configuration upload and download

Serial interface related information

The IPLink 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 IPLink serial ports are configured as DTE by default.

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

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

QoS related information

Check with your access service provider if there are any QoS related requirements, which you need to know prior to configuring IPLink software 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 IPLink software, Ethernet and serial ports represent the physical connectors on the IPLink hardware. Since ports are closely-knit with the physical structure of an IPLink, they cannot be created but have to be configured. The configuration of a port includes parameters for the physical and data link layer such as framing and encapsulation formats or media access control. Before any higher-layer user data can flow through a physical port, you must associate that port with an interface within the IP context. This association is referred to as a *binding*.

For information and examples on how to configure an Ethernet port, refer to chapter 13, "Ethernet port configuration" on page 137 or for a serial port to chapter 15, "Serial port configuration" on page 167.

Creating and configuring IP interfaces

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

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

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

For information and examples on how to create and configure an IP interface, refer to chapter 11, "IP interface configuration" on page 117.

Configuring NAPT

Network address port translation (NAPT), which is an extension to NAT, uses TCP/UDP ports in addition to network addresses (IP addresses) to map multiple private network addresses to a single outside address. NAPT enables small offices to save money by requiring only one official outside IP address to connect several hosts via an IPLink to the access network. Moreover, NAPT provides additional security, because the IP addresses of hosts attached via the IPLink are invisible to the external world. You can configure NAPT by creating a profile that is afterwards used on an explicit IP interface. In IPLink software terminology, an IP interface *uses* a NAPT profile, as shown in figure 17 on page 111.

For information and examples on how to configure NAPT refer to chapter 12, "NAT/NAPT configuration" on page 128.

Configuring static IP routing

IPLink software allows to define static routing entries, which are table mappings established by the network administrator prior to the beginning of routing. These mappings do not change unless the network administrator alters them. Algorithms that use static routes are simple to design and work well in environments in which network traffic is relatively predictable and where network design is relatively simple.

For information and examples on how to configure static IP routing, refer to chapter 17, "Basic IP routing configuration" on page 195.

Configuring RIP

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

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

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

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

For information and examples on how to configure Routing Information Protocol (RIP) refer to chapter 18, "RIP configuration" on page 201.

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, IPLink software 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 an IPLink. IPLink software tests packets against the conditions in an access list one by one. The first match determines whether IPLink software accepts or rejects the packet. Because IPLink software 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 19, "Access control list configuration" on page 211.

Configuring quality of service (QoS)

In IPLink software, the link scheduler enables the definition of QoS profiles for network traffic on a certain interface, as shown in figure 17 on page 111. 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

IPLink software QoS features described in chapter 14, “Link scheduler configuration” on page 148 address these diverse and common needs.

Chapter 11 IP interface configuration

Chapter contents

Introduction

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

Within IPLink software, 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 IPLink software. For higher layer protocols to become active, a physical port or circuit must be bound to an interface. IP interfaces can be bound physically to Ethernet, SDSL or Frame Relay ports according to the appropriate transport network layer.

Software IP interface configuration task list

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

- Creating an IP interface (see page 118)
- Deleting an IP interface (see page 119)
- Setting the IP address and netmask (see page 120)
- ICMP message processing (see page 121)
- ICMP redirect messages (see page 121)
- Router advertisement broadcast message (see page 121)
- Defining the MTU of the interface (see page 122)
- Configuring an interface as a point-to-point link (see page 123)
- Displaying IP interface information (see page 123)
- Testing connections with the **ping** command (see page 124)

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 you can enter specific configuration parameters for the IP interface *name*.

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.

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

Deleting an IP interface

Almost every configuration command has a **no** form. In general, use the **no** form to disable a feature or function. Use the command without the **no** keyword to re-enable a disabled feature or to enable a feature that is disabled by default.

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

Mode: Context IP

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

Example: Delete IP interfaces

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

List the existing interfaces:

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

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

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

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

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

Setting the IP address and netmask

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

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

The **ipaddress** command offers the following options:

unnumbered	Enables IP processing on an interface without assigning an explicit IP address to the interface.
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; i.e. a mask in the range 255.255.255.0 through 255.255.255.255.
dhcp	Enables the DHCP client on this interface. For more information on DHCP-client configuration refer to chapter 22, "DHCP configuration" on page 251.

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* for the IP interface *lan*, use the following commands in IP context configuration mode.

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

Configuring a NAPT DMZ interface

The NAPT allows one or more specific IP interfaces to be excluded from NAPT translations although their traffic is routed through an IP interface to which a NAPT profile is bound. This configuration is usually necessary, for DMZ networks connected to an Ethernet port, which uses public IP addresses.

Mode: interface ip <if-name>

Step	Command	Purpose
1	**[*name*] (if-ip)[**if-name**]# [no] napt-inside**	If *no napt-inside* is specified, the interface is excluded from NAPT. if however *napt-inside* is specified, the interface will be handled normally by the NAPT.

ICMP message processing

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

- ICMP redirect messages
- Router advertisement broadcast message

ICMP redirect messages

Routes are sometimes less than optimal. For example, the router may be forced to resend a packet through the same interface on which it was received. In this case, the IPLink software 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.

IPLink software ICMP message processing offers two options for host route redirects:

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

Mode: Interface

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

Example: ICMP redirect messages

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

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

Router advertisement broadcast message

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

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

Mode: Interface

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

Example: Router advertisement broadcast message

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

```
IPLink(ctx-ip)[router]#interface lan
IPLink(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 IPLink software application software will fragment any IP packet that exceeds the MTU set for an interface. The default MTU packet size is set to 1500 for an interface. In cases where fragmentation is not allowed along the IP connection, forcing a reduction of the MSS (*maximum segment size*) is the only viable solution.

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

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

Mode: Interface

Step	Command	Purpose
1	***node***(**ctx-ip)[router]#interface** *name*	Selects the interface *name* for ICMP message processing configuration
2	***node***(**if-ip)[***name***]#mtu size**	Sets the IP MTU packet size to *size* of the interface *name*. The MTU packet size value must be in the range from 48 to 1500.
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.

```
IPLink(ctx-ip)[router]#interface lan
IPLink(if-ip)[lan]#mtu 1000
IPLink(if-ip)[lan]#tcp adjust-mss rx mtu
IPLink(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 the interface ifname as point-to-point link

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

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

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

Displaying IP interface information

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

Mode: Operator execution or any administrator execution

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

Example: Displaying IP interface information

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

```
IPLink>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
…
```

Displaying dynamic ARP entries

The following command can be used to display the dynamically learned ARP entries on an IP interface or on the entire system.

Step	Command	Purpose
1	[*name*]**#show arp** [<ip-if-name>]	Display the ARP entries for the specified or all IP interfaces.

Flushing dynamic ARP entries

The following command can be used to flush the dynamically learned ARP entries on an IP interface or on the entire system.

Step	Command	Purpose
1	[*name*]**#arp flush**[<ip-if-name>]	Flushes the ARP entries for the specified or all IP interfaces.

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, you should first run it on the respective IPLink interface to verify that the local LAN or WAN interface is up and running. Then, you should “ping” hosts and gateways further away. Round-trip times and packet loss statistics are computed. If duplicate packets are received, they are not included in the packet loss calculation, although the round trip time of these packets is used to calculate the minimum/average/maximum round-trip time numbers. When five ICMP echo requests packets have been sent and received, a brief summary is displayed.

Example: Testing connections with the **ping** command

The following example shows how to invoke the echo protocol to the destination host at IP address 172.16.1.10 by using the **ping** command from operator execution mode.

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

Traceroute

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

Mode: Either operator or administrator execution

Step	Command	Purpose
1	***node*(cfg)# [no] debug arp**	Enables or disables the ARP debug monitor.
2	***node*(cfg)# show arp**	Summarizes the ARP information for each of the Ethernet ports.

Debug ARP

You may use the **debug arp** and **show arp** commands to assist you in debugging IP connectivity and its corresponding interfaces.

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

Example: Debug ARP output

```
IPLink(cfg)#debug arp
IPLink(cfg)#ping 10.9.10.11
Sending 5, 56 bytes, ICMP echo requests to 10.9.10.11:
17:25:40  ARP   > Entry 10.9.10.11: Sending first request
17:25:40  ARP   > Tx ARP Request: Who has 10.9.10.11 tell 10.9.10.1 at
00:A0:BA:00:92:4F
17:25:40  ARP   > Rx ARP Reply: 10.9.10.11 is at 00:50:04:74:94:6C tell 10.9.10.1 at
00:A0:BA:00:92:4F
17:25:40  ARP   > Entry 10.9.10.11: Updated by 00:50:04:74:94:6C
56 bytes from 10.9.10.11: Time 10ms
17:25:40  ARP   > Rx ARP Request: Who has 10.9.10.1 tell 10.9.10.3 at
00:09:5B:53:D2:B0
17:25:40  ARP   > Entry 10.9.10.3: Updated by 00:09:5B:53:D2:B0
17:25:40  ARP   > Tx ARP Reply: 10.9.10.1 is at 00:A0:BA:00:92:4F tell 10.9.10.3 at
00:09:5B:53:D2:B0
% Aborted
Ping statistics for 10.9.10.11:
  Packets: Sent 1, Received 1, Lost 0 (0% loss),
  RTT:     Minimum 10ms, Maximum 10ms, Average 10ms
```

Example: Display the ARP information.

```
IPLink(cfg)#show arp
IP Interface eth0:
-----------------------------------------------------------------------------
Remote IP        Remote MAC         State          TTL    TxReq  RxRep  Usage
-----------------------------------------------------------------------------
69.138.216.1     00:01:5C:22:46:C2  reachable       342s      2      2     12
-----------------------------------------------------------------------------

IP Interface eth1:
-----------------------------------------------------------------------------
Remote IP        Remote MAC         State          TTL    TxReq  RxRep Usage
-----------------------------------------------------------------------------
10.9.10.20       00:11:1A:4C:B1:1C  reachable       408s   1454   1451  67939
10.9.10.12       00:02:2D:BB:13:FB  reachable       326s    533    571  16819
10.9.10.2        00:09:5B:6F:93:06  reachable       518s      0    515   1054
10.9.10.166      00:09:5B:41:30:33  stale           556s      2      9   2277
10.9.10.10       00:80:AD:78:BB:DD  reachable       394s      0      2   1982
10.9.10.11       00:50:04:74:94:6C  reachable       433s      1      1      2
10.9.10.3        00:09:5B:53:D2:B0  reachable       521s      0      2     18
-----------------------------------------------------------------------------
```

Configuring the IGMP Proxy

To enable the IGMP proxy functionality, you need to define which interface shall be used to receive multicast streams (upstream interface) and to which interfaces the multicast streams shall be forwarded (downstream interfaces). The router then listens on the downstream interfaces for IGMP join messages and forwards them to the upstream interface.

Mode: Context IP

Step	Command	Purpose
1	***node*(ctx-ip)[ctx-*name*]# interface <if-name>**	Go to the IP interface, which shall act as the IGMP proxy upstream interface
2	**node(if-ip)[if-*name*]# igmp interface-type proxy-upstream**	Define the interface as the IGMP proxy upstream interface
3	***node*(ctx-ip)[ctx-*name*]# interface <if-name>**	Go to an IP interface, which shall act as an IGMP proxy downstream interface
4	***node*(if-ip)[if-*name*]# igmp interface-type proxy-downstream**	Define the interface as an IGMP proxy downstream interface
5		Repeat steps 3 & 4 for any additional interface, which shall act as an IGMP proxy downstream interface.

Examples

Deleting an IP interface

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

List the existing interfaces in the IP context:

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

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

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

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

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

Chapter 12 NAT/NAPT configuration

Chapter contents

Introduction

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

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

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

For further information about the functionality of Network Address Translation (NAT) and Network Address Port Translation (NAPT), consult the RFCs 1631 and 3022. This chapter applies the terminology defined in RFC 2663.

IPLink software provides four types of NAT/NAPT:

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

You can combine these types of NAT/NAPT without any restriction. One type of profile, the 'NAPT Profile', holds the configuration information for all four types where configuration is required. The remainder of this Section shortly explains the behavior of the different NAT/NAPT types.

Dynamic NAPT

Dynamic NAPT is the default behavior of the NAT/NAPT component. It allows hosts on the local network to access any host on the global network by using the global interface address as source address. It modifies not only the source address, but also the source port, so that it can tell different connections apart (NAPT source ports are in the range 8,000 to 16,000). UDP and TCP connections from the local to the global network trigger the creation of a dynamic NAPT entry for the reverse path. If a connection is idle for some time (UDP: 2 minutes, TCP: 12 hours) or gets closed (only TCP), the dynamic NAPT entry is removed.

An enhancement of the Dynamic NAPT allows to define subsets of hosts on the local network that shall use different global addresses. Up to 20 subsets with their respective global addresses are possible. Such a global NAPT address can be any IP address as long as the global network routes the traffic to the global interface of the NAT/NAPT component.

Note Only the NAT/NAPT component handles global NAPT addresses. Other components of the IPLink are not accessible via these addresses.

Figure 18 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.

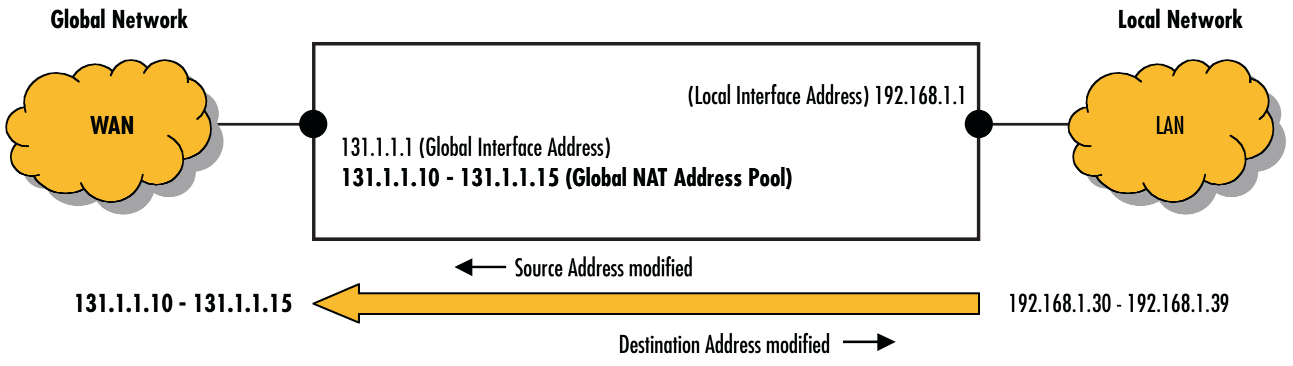

Global Network
Local Network
WAN
LAN
(Local Interface Address) 192.168.1.1
131.1.1.1 (Global Interface Address)
131.1.1.10 - 131.1.1.15 (Global NAT Address Pool)
Source Address modified
131.1.1.10 - 131.1.1.15
192.168.1.30 - 192.168.1.39
Destination Address modified

Figure 18. 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.

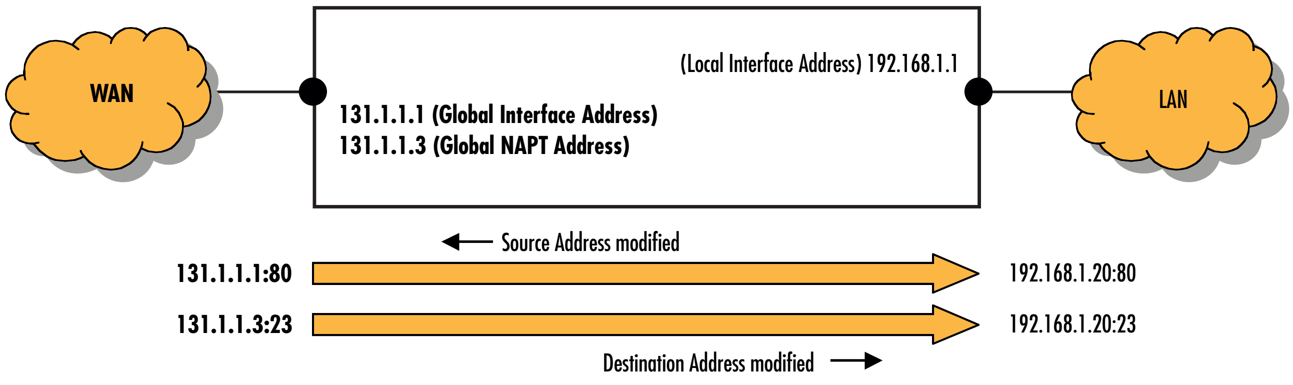

WAN
LAN
(Local Interface Address) 192.168.1.1
131.1.1.1 (Global Interface Address)
131.1.1.3 (Global NAPT Address)
Source Address modified
131.1.1.1:80
192.168.1.20:80
131.1.1.3:23
192.168.1.20:23
Destination Address modified

Figure 19. Static NAPT

Note Be careful when mapping ports the IPLink uses itself (e.g. Telnet, TFTP) because the IPLink might become inaccessible.

Dynamic NAT

NAT only modifies addresses but not ports. Dynamic NAT assigns a global address from a global NAT address pool each time a local host wants to access the global network. It creates a dynamic NAT entry for the reverse path. If a connection is idle for some time (2 minutes), the dynamic NAT entry is removed. Should Dynamic NAT run out of global addresses, it lets Dynamic NAPT handle the connection (which may lead to an unexpected behavior).

Dynamic NAT is particularly useful for protocols that do not build on UDP or TCP but directly on IP (e.g. GRE, ESP). See also section "NAPT traversal" on page 131.

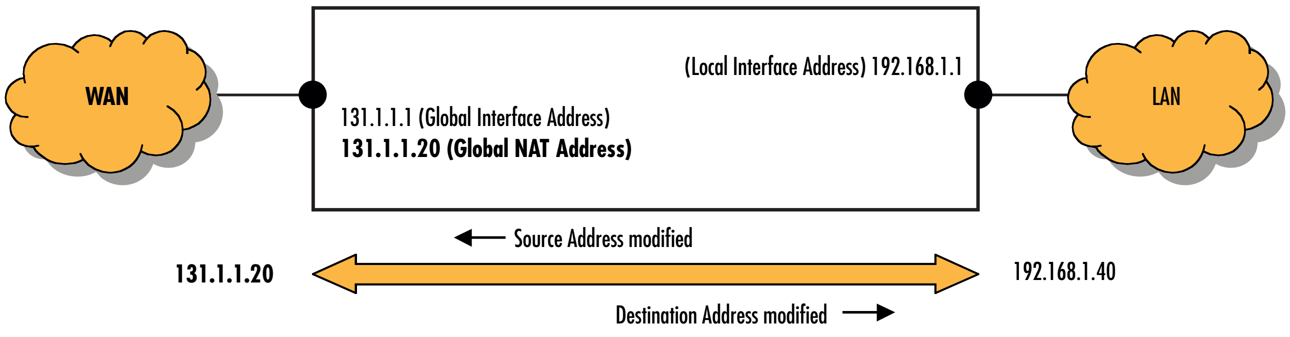

WAN
(Local Interface Address) 192.168.1.1
131.1.1.1 (Global Interface Address)
131.1.1.20 (Global NAT Address)
LAN
Source Address modified
131.1.1.20
192.168.1.40
Destination Address modified

Figure 20. 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 IPLink itself.

Static NAT is particularly useful for protocols that do not build on UDP or TCP but directly on IP (e.g. GRE, ESP). See also section "NAPT traversal" on page 131.

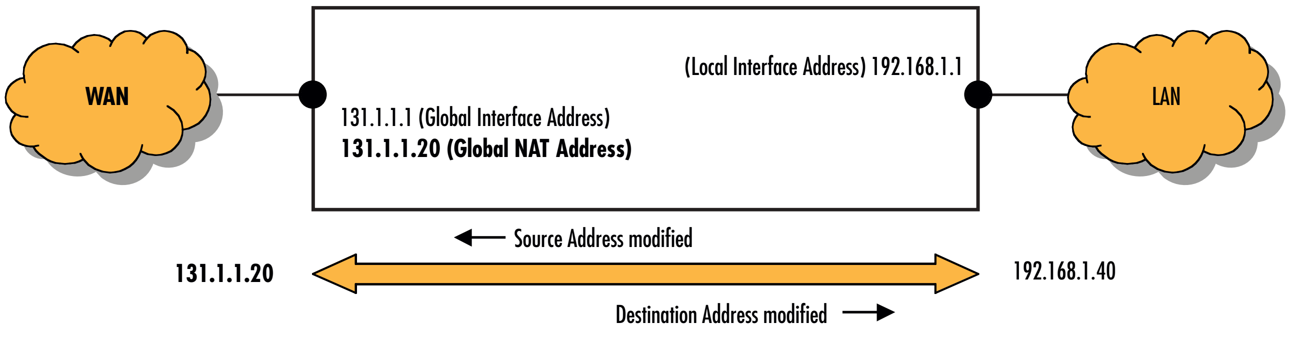

WAN
(Local Interface Address) 192.168.1.1
131.1.1.1 (Global Interface Address)
131.1.1.20 (Global NAT Address)
LAN
Source Address modified
131.1.1.20
192.168.1.40
Destination Address modified

Figure 21. 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), do not easily traverse a NAPT.

The IPLink software 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 IPLink software 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 132)
- Activating NAT/NAPT (see page 132)
- Displaying NAT/NAPT configuration information (see page 135)

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

Step	Command	Purpose
1	*node*(cfg)#**profile napt** *name*	Creates the NAPT profile *name* and activates the basic behavior of the Dynamic NAPT
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.
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	Command	Purpose
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.
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)
6 (optional)	*node*(pf-napt)[*name*]#**static { ah|esp|gre|ipv6 }** *local_ip* **[***global_ip***]**.	Creates a static NAT entry: traffic of the IP protocol AH, ESP, GRE, or IPv6 respectively directed to the *global_ip* is forwarded to the *local_ip*.

Use **no** in front of the above commands to delete a specific entry or the whole profile.

Note The command **icmp default** is obsolete.

Example: Creating a NAPT Profile

The following example shows how to create a new NAPT profile *access* that contains all settings necessary to implement the examples in section "Introduction" on page 129.

```
IPLink(cfg)#profile napt access
IPLink(pf-napt)[access]#range 192.168.1.10 192.168.1.19 131.1.1.2
IPLink(pf-napt)[access]#static tcp 192.168.1.20 80
IPLink(pf-napt)[access]#static tcp 192.168.1.20 23 131.1.1.3
IPLink(pf-napt)[access]#range 192.168.1.30 192.168.1.39 131.1.1.10 131.1.1.15
IPLink(pf-napt)[access]#static 192.168.1.40 131.1.1.20
IPLink(pf-napt)[access]static ah 192.168.1.41 131.1.1.120
```

Configuring a NAPT DMZ host

The NAPT allows a DMZ host to be configured, which receives any inbound traffic on the global NAPT interface, which:

- Is not translated by any static or dynamic NAPT entry and
- Is not handled by the device itself.

The following procedure shows how a DMZ host can be configured.

Mode: profile napt <pf-name>

Step	Command	Purpose
1	[*name*] **(pf-napt)**[pf-name]**# [no] dmz-host <dmz-host-ip-address> [<global-ip-address>]**	Configures a DMZ host. The global-ip-address must only be specified, if the DMZ host shall handle the inbound traffic for a different NAPT global IP address than the gateways global interface IP address.

Defining NAPT port ranges

The TCP/UDP port ranges to be used by the NAPT can be defined using the following procedure. The default port ranges for both TCP/UDP are 8000 to 15999.

Mode: profile napt <pf-name>

Step	Command	Purpose
1	[*name*] **(pf-napt)**[pf-name]**# tcp-port-range <range-start-tcp-port> <range-end-tcp-port>**	Define the TCP port range
2	[*name*] **(pf-napt)**[pf-name]**# udp-port-range <range-start-udp-port> <range-end-udp-port>**	Define the UDP port range

Preserving TCP/UDP port numbers in NAPT

The NAPT can be configured to preserve the TCP/UDP port number of outbound packets sent from local hosts towards the global NAPT interface. If this option is enabled the NAPT tries not to change these port numbers. If the port is however already in use, the NAPT will ignore this setting and assign a port number from the configured TCP/UDP port ranges.

Mode: profile napt <pf-name>

Step	Command	Purpose
1	[*name*] **(pf-napt)**[pf-name]**# [no] preserve-tcp-ports**	Enable/disable preserving of TCP ports.
2	[*name*] **(pf-napt)**[pf-name]**# [no] preserve-udp-ports**	Enable/disable preserving of UDP ports.

Defining the UDP NAPT type

The NAPT type to be applied for UDP packets is configurable using the following procedure. The NAPT supports the UDP translation types shown in the following list. The list is ordered by the security of the NAPT type starting with the highest security type.

- symmetric
- port-restricted-cone
- address-restricted-cone
- full-cone

You find a detailed description of these NAPT types in section 5 of RFC3489. To allow STUN to work through the NAPT the *full-cone setting* is usually required. The default setting is *symmetric*.

Mode: profile napt <pf-name>

Step	Command	Purpose
1	[*name*] **(pf-napt)**[pf-name]**# udp-handling {symmetric|address-restricted-cone|port-restricted-cone|full-cone}**	Define the UDP translation type

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*.

```
IPLink(cfg)#context ip router
IPLink(ctx-ip)[router]#interface lan
IPLink(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

Step	Command	Purpose
1	*node*(cfg)#**show profile napt**	Displays the available NAPT profiles
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

```
IPLink(pf-napt)[access]#show profile napt access
NAPT profile access:
--------------------------
STATIC NAPT MAPPINGS
  Protocol     Local IP             Local Port       Global IP        Global Port
  --------    ---------------      -----------      ---------------   -----------
  tcp          192.168.1.20         80               0.0.0.0           80
  tcp          192.168.1.20         23               131.1.1.3         23

STATIC NAT PROTOCOL MAPPINGS
  Protocol Local IP        Global IP
  -------- --------------- ---------------
  ah       192.168.1.41    131.1.1.120

STATIC NAT MAPPINGS
  Local IP        Global IP
  --------------- ---------------
  192.168.1.40    131.1.1.20

STATIC NAPT RANGE MAPPINGS
  Local IP Start  Local IP Stop   Global IP
  --------------- --------------- ---------------
  192.168.1.10    192.168.1.19    131.1.1.15

STATIC NAT RANGE MAPPINGS
  Local IP Start  Local IP Stop   Global IP Start Global IP Stop
  --------------- --------------- --------------- ---------------
  192.168.1.30    192.168.1.39    131.1.1.10      131.1.1.15
```

Configuring NAT static protocol entries

The following command adds a static NAT entry, which causes any packets of the specified protocol received on the global side of the NAT to be forwarded to the host specified on the local side of the NAT.

```
node(pf-napt)[ name]#static { udp | tcp } local-ip local-port [ global-ip] [ global-port]
```

Mode: profile napt <pf-napt>

Step	Command	Purpose
1	[*name*](**pf-napt**)# **static <protocol> <local-ip-address> [<global-ip-address>]**	Adds a static NAT protocol entry

Chapter 13 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 IPLink software.

For IPLink 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 defaults for the Ethernet ports are as follows:

- Medium auto
- Encapsulation ip
- Bind interface eth1 router (for port Ethernet 0 1) and bind interface eth0 router (for port Ethernet 0 0)
- Enabled.

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 139)
- Configuring medium for an Ethernet port (see page 139)
- Configuring Ethernet encapsulation type for an Ethernet port (see page 140)
- Binding an Ethernet port to an IP interface (see page 140)
- Configuring multiple IP addresses on the Ethernet ports (see page 141)
- Configuring a VLAN (see page 142)
- Configuring layer 2 CoS to service-class mapping for an Ethernet port (advanced) (see page 143)
- Closing an Ethernet port (see page 145)

Entering the Ethernet port configuration mode

To enter port configuration mode and begin configuring an Ethernet port, enter the command **port ethernet** *slot port* in administrator execution mode. The keywords *slot* and *port* represent the number of the respective physical entity as show in table 6.

Table 6. Permanent built-in interface slot and port mapping for IPLink Series

Device Model	Interface type	Slot	Port	Interface
IPLink 2802	Ethernet (WAN)	0	0	eth0
	Ethernet (LAN)	0	1	eth1
IPLink 2805	Ethernet (WAN)	0	0	eth0
	Ethernet (LAN)	0	1 - 4	eth0 - eth4
IPLink 2803	Ethernet (WAN)	0	0	eth0
	Ethernet (LAN)	0	1	eth1
	Serial WAN			e1t1
IPLink 2821	Ethernet (WAN)	0	0	eth0
	Ethernet (LAN)	0	1	eth1
	Serial WAN			X.21
IPLink 2835	Ethernet (WAN)	0	0	eth0
	Ethernet (LAN)	0	1	eth1
	Serial WAN			V.35

Since a port must be configured unambiguously, choose the appropriate expansion slot and port number. The number and type of available ports depend upon your IPLink model, and also on the interface card fit for IPLink series devices. All permanent on-board interfaces of an IPLink 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 the interface on *slot* and *port* to medium according to the selected option.

Note The following restrictions apply:

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

Example: Configuring medium for an Ethernet port

The following example shows how to configure medium auto-sense for the Ethernet port on slot 0 and port 0 of an IPLink device.

```
IPLink(cfg)#port ethernet 0 0
IPLink(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 an IPLink series device.

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

Binding an Ethernet port to an IP interface

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

Figure 22 shows the logical binding of the Ethernet port at slot *0* on port *0* to the IP interface *lan* which is defined in the IP context router.

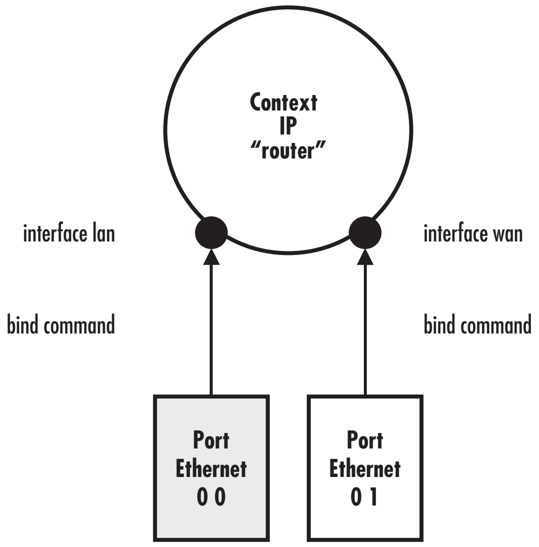


Figure 22. Binding of an Ethernet port to an IP interface

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 an IPLink device to an already existing IP interface *lan*.

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

Multiple IP addresses on Ethernet ports

It is possible to use multiple IP addresses on an Ethernet port by binding the port to multiple IP interfaces. Each of the IP interfaces uses an IP address of one of the subnets on the Ethernet ports.

The procedures below demonstrate how to IP addresses of two different networks can be used on an Ethernet port. However, if necessary any number of IP interfaces can be bound to an Ethernet port.

Mode: Configure

Step	Command	Purpose
1	[*name*] **(cfg)# context ip**	Enter the IP context configuration mode.
2	[*name*] **(ctx-ip)**[router]**# interface** <ip-if-1-name>	Create the first IP interface.
3	**[name] (if-ip)[<ip-if-1-name>]# ipaddress** <ip-address-1> <subnet-mask-1>	Set the IP address for the first IP interface
4	[*name*] **(if-ip)[** <ip-if-1-name>**]# interface** <ip-if-2-name>	Create the second IP interface.
5	[*name*] **(if-ip)[** <ip-if-2-name>**]# ipaddress** <ip-address-2> <subnet-mask-2>	Set the IP address for the second IP interface
6	[*name*] **(if-ip)[** <ip-if-2-name>**]# port ethernet** <slot> <port>	Enter Ethernet port configuration mode
7	[*name*] **(prt-eth)[**<slot>/<port>**]# encapsulation ip**	Set the encapsulation to IP
8	[*name*] **(prt-eth)[**<slot>/<port>**]# bind interface <ip-if-1-name>**	Bind the port to the first IP interface
9	[*name*] **(prt-eth)[**<slot>/<port>**]# bind interface <ip-if-2-name>**	Bind the port to the second IP interface
10	[*name*] **(prt-eth)[**<slot>/<port>**]# no shutdown**	Enable the Ethernet port

Configuring a VLAN

By default no VLAN ports are configured on an Ethernet port. One or more VLAN ports can be created on each Ethernet port.

You must bind the VLAN port to an existing IP interface. When executing the **bind** command, the requested interface must exist.

For incoming VLAN packets each of the 8 possible layer 2 class of services (CoS) can be mapped to a traffic class. Unless otherwise specified all CoS values map to the default traffic class.

By default all VLAN ports are initially disabled. They can be enabled with the **no shutdown** command. The corresponding Ethernet port must also be enabled for the VLAN port to work. If the Ethernet port is disabled, all associated VLAN ports are also disabled.

When a VLAN port is closed, the IP interface that is bound to this port is also closed. All static routing entries that are using this interface change their state to *invalid* and all dynamic routing entries will be removed from the route table manager.

Mode: Configure

Step	Command	Purpose
1	***node*(config)#port ethernet** *slot port*	Enter Ethernet port configuration.
2	***node*(prt-eth)[*slot/port*]#vlan** *id*	Create new VLAN port.
3	***node*(vlan)[*id*]#encapsulation {ip|pppoe|multi}**	Defines the payload type(s) to be used on this VLAN: • **ip**: IP traffic only (not used for PPP) • **pppoe**: PPPoE sessions only • **multi**: both IP traffic and PPPoE sessions For more information on the PPP/PPPoE configuration see chapter 25, "PPP configuration" on page 270.
4	***node*(vlan)[*id*]#bind interface** *name* **[router]**	Bind the VLAN port to the existing interface name. If no IP context is given, the system attaches the interface to the default IP context known as router.
5	***node*(vlan)[*id*]#map cos** *layer-2-CoS-value* **to** *traffic-class-name*	Selects the layer 2 CoS (Class of Service) to traffic class mapping. The traffic class must already exist.
6	***node*(vlan)[*id*]#no shutdown**	Activate the VLAN port.
7	***node*(vlan)[*id*]#exit *node*(prt-eth)[*slot/port*]# no shutdown**	Make sure the hosting Ethernet port is also activated.

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

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

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

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

Mode: Configure

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

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

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

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

Do the following to configure the class of service map:

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

Adding a receive mapping table entry

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

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

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#port ethernet** *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 an IPLink.

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

Adding a transmit mapping table entry

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

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

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#port ethernet** *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 an IPLink.

```
IPLink(cfg)#port ethernet 0 0
IPLink(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 to open the port with the interface to which it is bound.

Example: Disabling an Ethernet port

The following example shows how to disable the Ethernet port on slot 0 and port 0 of an IPLink device.

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

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

```
IPLink(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.

```
IPLink(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
…
```

Using the built-in Ethernet sniffer

The software contains a built-in sniffer, which can be used to capture data packets on Ethernet ports. The sniffer saves the captured data to a file in the systems flash file system. The file can later be uploaded via TFTP for viewing. The files can be viewed with many sniffer applications, for example, Ethereal. The capture buffer can hold a maximum of 1000 packets or 100kByte of data.

The sniffer is controlled via the following CLI command:

	Command	Purpose
	[*name*] (cfg)# [no] sniff ethernet <slot> <port> [wrap]	Enable/disable the sniffer

The following is an example of how the sniffer is normally used:

Step	Command	Purpose
1	[*name*] **(cfg)# sniff ethernet 0 1 [wrap]**	Enable the sniffer on ethernet port 0 1. (Normally the sniffer stops capturing, if the capture buffer is full. However, if the 'wrap' option is specified, the sniffer starts discarding the oldest packets and retains the newest ones, if the capture buffer is full.)
2		Now the sniffer is active and will capture the datapackets on the specified ethernet port.
3	[*name*] **(cfg)# no sniff ether-net 0 1]**	Disable the sniffer on ethernet port 0 1. (Note, that the captured data is not stored to flash memory unless you issue this command) The file in the flash memory will be named as follows: *nvram:ethernet-0-<slot>-<port>.cap* In this example the name will be: *nvram:ethernet-0-0-1.cap*
4	[*name*] **(cfg)# copy nvram:ethernet-0-0-1.cap tftp://tftp.mypc.net/ capture.cap**	Copy the capture file via TFTP to a workstation.
5	[*name*] **(cfg)# erase nvram:ethernet-0-0-1.cap**	Erase the capture file on the system to save flash memory.
6		Now the capture file capture.cap can be viewed on a workstation with Ethereal for example.

Note It is possible to capture packets on multiple Ethernet ports at the same time.

Chapter 14 Link scheduler configuration

Chapter contents

Introduction

This chapter describes how to use and configure the IPLink software Quality of Service (QoS) features. Refer to chapter 19, "Access control list configuration" on page 211 more information on the use of access control lists.

This chapter includes the following sections:

- Quick references (see page 152)
- Packet Classification (see page 154)
- Assigning bandwidth to traffic classes (see page 152)
- Link scheduler configuration task list (see page 153)

QoS in networking refers to the capability of the network to provide a better service to selected network traffic. In the context of VoIP, the primary issue is to control the coexistence of voice and data packets such that voice packets are delayed as little as possible. This chapter shows you how to configure IPLink software to best use the access link.

In many applications you can gain a lot by applying the minimal configuration found in the quick reference section, but read sections "Applying scheduling at the bottleneck" and "Using traffic classes" first to understand the paradox of why we apply a rate-limit to reduce delay and what a "traffic-class" means.

Applying scheduling at the bottleneck

When an IPLink acts as an access router and voice gateway, sending voice and data packets to the Internet, the access link is the point where intelligent use of scarce resources really makes a difference. Frequently, the access link modem is outside of the IPLink and the queueing would happen in the modem, which does distinguish between voice and data packets. To improve QoS, you can configure the IPLink to send no more data to the Internet than the modem can carry. This keeps the modem's queue empty and gives the IPLink software control over which packet is sent over the access link at what time.

Using traffic classes

The link scheduler needs to distinguish between different types of packets. We refer to those types as "traffic-classes". You can think of the traffic-class as if every packet in the IPLink has a tag attached to it on which the classification can be noted. The access control list "stage" (ACL) can be used to apply such a traffic-class name to some type of packet based on its IP-header filtering capabilities. The traffic-class tags exist only inside the IPLink, but layer 2 priority bits (802.1pq class-of-service) and IP header type-of-service bits (TOS field) can be used to mark a specific packet type for the other network nodes. By default the traffic-class tag is empty. Only two types of packets are automatically marked by the IPLink software: voice packets and data packets origination from or destined to the IPLink itself are marked as "local-voice" and "local-default" respectively. Please refer to figure 23 on page 150 when using the ACL to classify traffic. It illustrates the sequence of processing stages every routed packet passes. Only stages that have been installed in the data path with a "use profile..." statement in the corresponding interface configuration are present. Both an input direction ACL on the receiving interface as well as an output ACL on the transmitting interface can be used to classify a packet for special handling by the output link scheduler on the transmit interface. But as visible from the figure no ACL can be used for an input link scheduler.

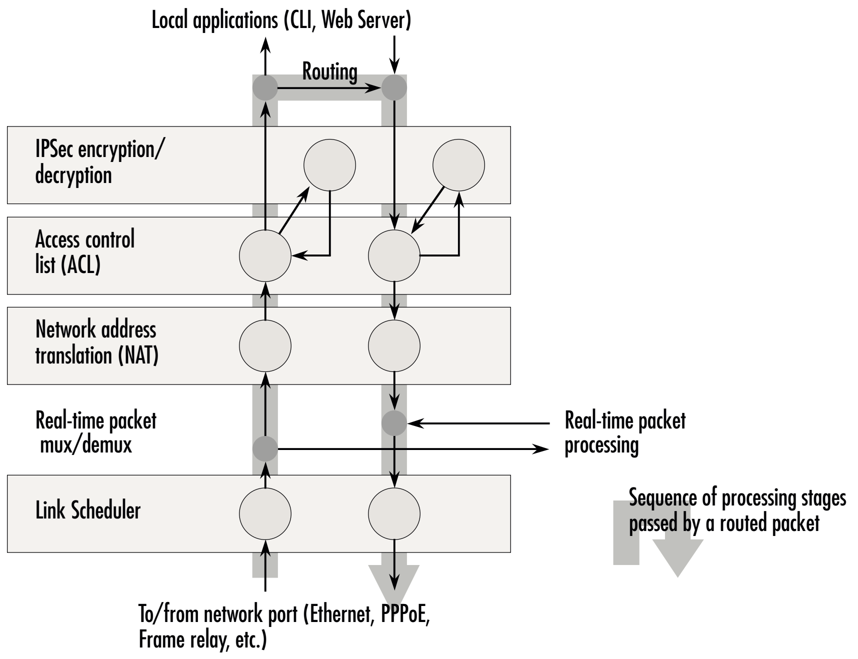

Local applications (CLI, Web Server)
Routing
IPSec encryption/ decryption
Access control list (ACL)
Network address translation (NAT)
Real-time packet mux/demux
Real-time packet processing
Link Scheduler
Sequence of processing stages passed by a routed packet
To/from network port (Ethernet, PPPoE, Frame relay, etc.)

Figure 23. Packet routing in IPLink software

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

Introduction to Scheduling

Scheduling essentially means to determine the order in which packets of the different traffic-classes are served. The following sections describe the ways this arbitration can be done.

Priority

One way of ordering packets is to give priority to one traffic-class and to serve the other traffic-classes when the first has nothing to send. IPLink software uses the priority scheme to make sure that voice packets generated by the IPLink will experience as little delay as possible. Voice packets can receive this treatment because they will not use up the entire bandwidth.

Weighted fair queuing (WFQ)

This arbitration method assures a given minimal bandwidth for each source. An example: you specify that traffic-class A gets three times the bandwidth of traffic-class B. So A will get a minimum of 75% and B will get a minimum of 25% of the bandwidth. But if no class A packets are waiting B will get 100% of the bandwidth. Each traffic-class is in fact assigned a *relative* weight, which is used to share the bandwidth among the currently active classes. Patton recommends that you specify the weight as percent which is best readable.

Shaping

There is another commonly used way to assign bandwidth. It is called *shaping* and it makes sure that each traffic-class will get just as much bandwidth as configured and not more. This is useful if you have subscribed to a service that is only available for a limited bandwidth e.g. low delay. When connecting the IPLink to a *DiffServ* network shaping might be a required operation.

Burst tolerant shaping or wfq

For weighted fair queuing and shaping there is a variation of the scheduler that allows to specify if a traffic class may temporarily receive a higher rate as long as the average stays below the limit. This burstiness measure allows the network to explicitly assign buffers to bursty sources.

When you use shaping on the access link the shaper sometimes has the problem that multiple sources are scheduled for the same time - and therefore some of them will be served too late. If the rate of every source had to strictly obey its limit, all following packets would also have to be delayed by the same amount, and further collisions would reduce the achieved rate even further. To avoid this effect, the IPLink software shaper assumes that the burstiness needed for sources to catch up after *collisions* is implicitly allowed. Future versions of IPLink software might allow setting the burst rate and bursting size if more control over its behavior is considered necessary.

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

Hierarchy

An arbiter can either use wfq *or* shaping to determine which source to serve next. If you want the scheduler to follow a combination of decision criteria you can combine different schedulers in hierarchy to do a multi-level arbitration. Hierarchical scheduling is supported in IPLink software with service-policy profiles used inside service-policy profiles.In figure 24 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.

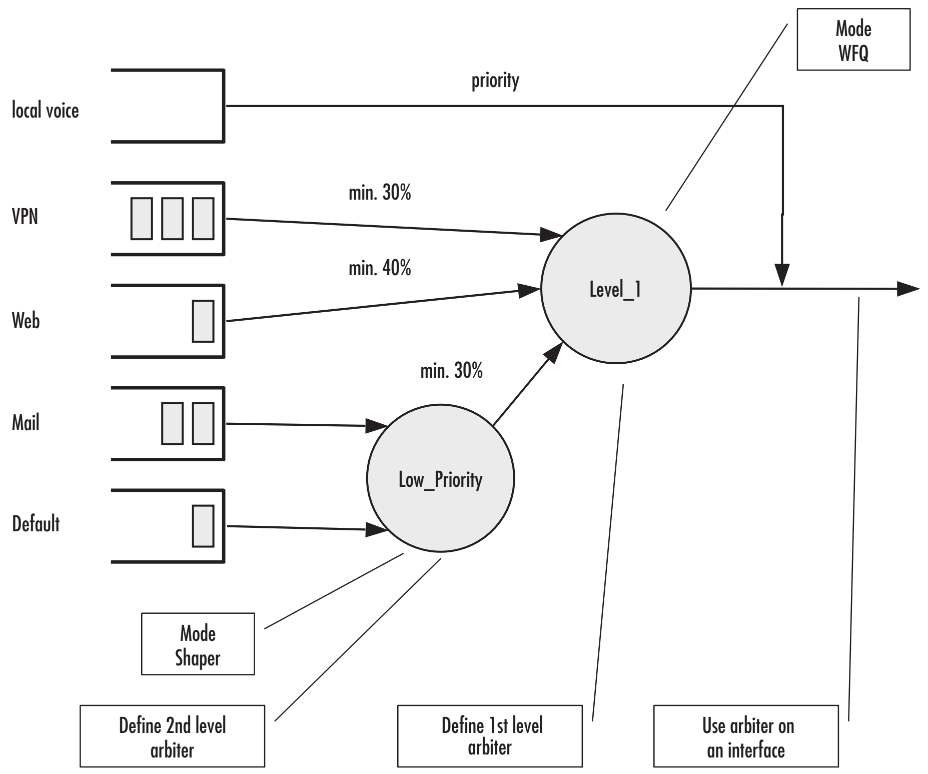


Figure 24. Example of Hierarchical Scheduling

Quick references

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

Setting the modem rate

To match the voice and data multiplexing to the capacity of the access link is the most common application of the IPLink software link scheduler.

1. Create a minimal profile.

```
profile service-policy modem-512
  rate-limit 512 header-length 20 atm-modem
  source traffic-class local-voice
    priority
```

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

```
context ip
interface wan
  use profile service-policy modem-512 out
```

Some explanations:

- “modem-512” is the title of the profile which is referred to when installing the scheduler
- “rate-limit 512” allows no more than 512 kbit/sec to pass which avoids queueing in the modem.
- “header-length 20” specifies how many framing bytes are added by the modem to “pack” the IP packet on the link. The framing is taken into account by the rate limiter.
- “atm-modem” tells the rate limiter that the access link is ATM based. This option includes the ATM overhead into the rate limit calculation. Please add 8 bytes to the header-length for AAL5 in this case.
- “source traffic-class” enters a sub-mode where the specific handling for a traffic-class is described. The list of sources in the service-policy profile tells the arbiter which “traffic sources” to serve.
- “local-voice” is the predefined traffic-class for locally terminated voice packet streams.
- “priority” means that packet of the source being described are always passed on immediately, packets of other classes follow later if the rate limit permits.

Command cross reference

Comparing IPLink software with the Cisco IOS QoS software command syntax often helps administrators to straightforwardly configure IPLink devices. In table 7 the Cisco IOS Release 12.2 QoS commands are in contrast with the respective IPLink software commands.

Table 7. Command cross reference

Action	IOS command	IPLink software command
Specifies the name of the policy map or profile to be created or modified.	**policy-map** policy-map-name	**profile service-policy** *profile-name*
Specifies the name of the class map or class to be created.	**class-map** *class-map-name*	**source traffic-class** *class-name*
For IOS specifies average or peak bit rate shaping. For IPLink software 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 IPLink software assigns the weight of the selected source (only used with wfq).	**bandwidth** {*bandwidth-kbps* | **percent** *percent*}	**share** *percent-of-bandwidth*

Link scheduler configuration task list

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

- Defining the access control list profile
- Creating a service-policy profile (see page 156)
- Specifying the handling of traffic-classes (see page 158)
- Devoting the service policy profile to an interface (see page 164)

- Displaying link arbitration status (see page 165)
- Displaying link scheduling profile information (see page 165)
- Enable statistics gathering (see page 165)

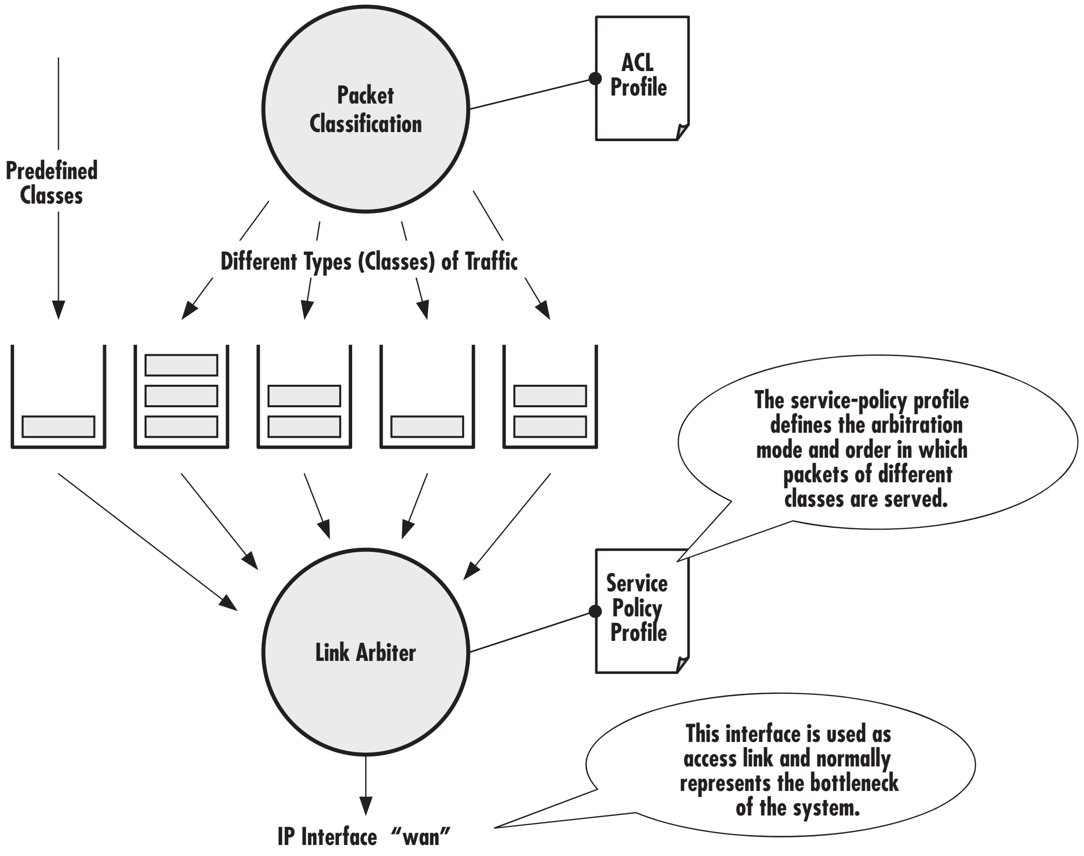

Packet Classification
ACL Profile
Predefined Classes
Different Types (Classes) of Traffic
The service-policy profile defines the arbitration mode and order in which packets of different classes are served.
Service Policy Profile
Link Arbiter
This interface is used as access link and normally represents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

Figure 25. Elements of link scheduler configuration

Defining the access control list profile

Packet classification

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

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

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

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

Creating an access control list

The procedure to create an access control list is described in detail in chapter 19, "Access control list configuration" on page 211.

At this point a simple example is given, that shows the necessary steps to tag any outbound traffic from a Web server. The scenario is depicted in figure 26. 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.

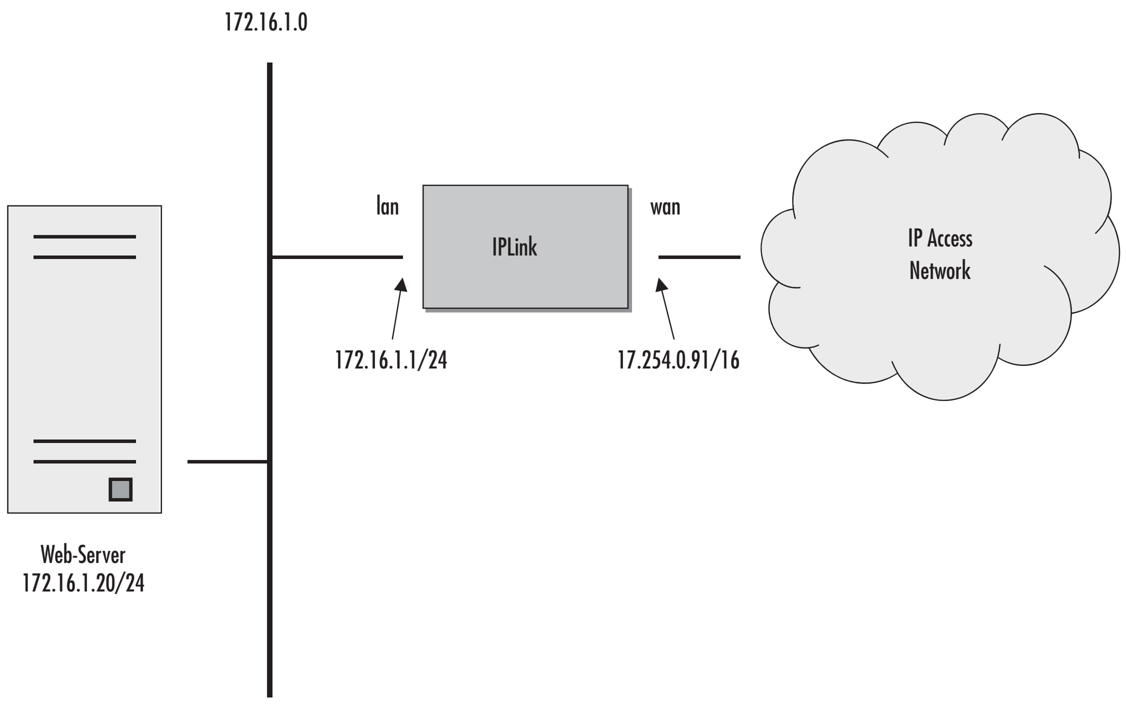


Figure 26. Scenario with Web server regarded as a single source host

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

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

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

Mode: Configure

Step	Command	Purpose
1	***node***(**cfg)#profile acl** *name*	Creates a new access control list profile named *name*
2	***node***(**pf-acl)[***name***]#permit ip host ip-address any traffic-class** *class-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 traffic-class *class-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 traffic-class *Web* is added. Finally an IP access control list entry that permits IP traffic to or from all IP addresses is added to the access control list.

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

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

Creating a service policy profile

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

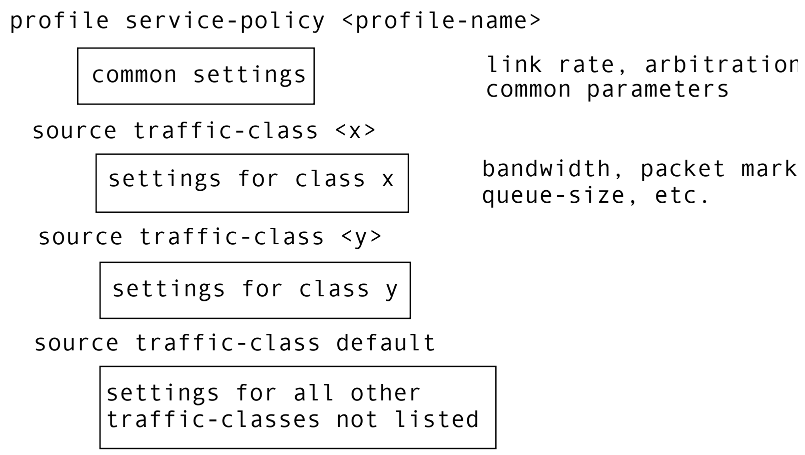


Figure 27. Structure of a Service-Policy Profile

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

Example: Creating a service policy profile

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

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

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

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

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

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

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

Mode: Configure

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

Specifying the handling of traffic-classes

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

Defining fair queuing weight

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

Mode: Source

Command	Purpose
node(**src**)[*name*]**#share** *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 no more), when overloaded the shaper will behave like a wfq arbiter. Bit-rate specification for shaper (kilobits).

Mode: Source

Command	Purpose
node(**src)[name]#rate** [*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.

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).

Table 8. TOS values and their meaning

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

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

Mode: Source

Command	Purpose
***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*. Standard ToT values are 0, 1, 2, 4, and 8, as given in table 8 on page 160, but any number from 0 to 15 can be configured.

Specifying the precedence field

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

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

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

Mode: Source

Command	Purpose
***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 (DSCP) marking

Differentiated services enhancements to the Internet protocol are intended to enable the handling of "traffic-classes" throughout the Internet. In this context the IP header TOS field is interpreted as something like a "traffic-class" number called. With IPLink software you can inspect the DSCP value in the ACL rules and modify the DSCP value with the link scheduler **set ip dscp** command.

Note When configuring service differentiation on the IPLink, ensure that code-point settings are arranged with the service provider.

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

Mode: Source

Command	Purpose
***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 9 lists the tag components.

Table 9. Traffic control info (TCI) field

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

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

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

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

Mode: Source

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

Mode: Source

Command	Purpose
***node*(src)[*name*]#random-detect {*burst-tolerance*}**	Defines random early detection (RED) for queues of for the selected traffic-class or policy *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.

Quality of Service for routed RTP streams

IPLink software supports including routed RTP packets in the QoS process. This is possible for plain streams as well as for encrypted streams in up- and downlink direction. The identification of the packets that have to be included in the QoS process base upon their size. In the service-policy profile exists a command that allows mapping of a specific packet size or a range to a traffic class.

There are two predefined ranges the user can choose. One of them is 'routed-voice' that specifies a packet size range from 50 Byte to 280 Byte the other one is 'routed-voice-encrypted' that specifies a packet size range from 92 Byte to 324 Byte. By selecting this predefined ranges all voice packets from G.729/10ms to G.711/30ms will be assigned to the configured traffic-class.

Be aware that also other packets matching the configured size or range will be assigned to the specified traffic-class. All values to be configured are in Byte and are IP Packet sizes (IP Header plus Payload).

Mode: profile service-policy/profile

Command	Purpose
[*name*] **(pf-srvp)**[<name>]**# [no] map packet-size {routed-voice | routed-voice-encrypted | [**<lower-size> <upper-size>**] } traffic-class** <traffic-class-name>	Assigns IP packets of a predefined or specified range to a traffic-class. To name a specific size, configure lower-range and upper-range with the same value.

The following procedure guides through the steps required for creating, configuring and using service policy profiles on a WAN link that has an upstream and downstream capacity of 256kBit/s and is based on ADSL technology. The access device must be able to process the RTP traffic generated by a VoIP Phone located in the LAN like the local generated RTP stream.

Mode: Configure

Step	Command	Purpose
1	[*name*] **(cfg)# profile service-policy** <name-out>	Creates a new service policy profile will be configured for the uplink.
2	[*name*] **(pf-srvp)**[<name-out>]**# rate-limit 256 atm-modem**	Configures the uplink capacity.
3	[*name*] **(pf-srvp)**[<name-out>]**# map packet-size routed-voice traffic-class local-voice**	Specifies that routed voice traffic will be processed like local generated voice traffic.
4	[*name*] **(pf-srvp)**[<name-out>]**# source traffic-class local-voice**	Enters traffic-class configuration mode
5	[*name*] **(src)**[local-v~]**# priority**	Specifies that local-voice has priority. Because route-voice is mapped to local-voice, also routed-voice has priority.
6	[*name*] **(src)**[local-v~]**# profile service-policy** <name-in>	Creates a new service policy profile will be configured for the downlink.
7	[*name*] **(pf-srvp)**[<name-in>]**# rate-limit 256 atm-modem voice-margin 80**	Configures the downlink capacity and sets a voice-margin of 80kBit/s
8	[*name*] **(pf-srvp)**[<name-in>]**# map packet-size routed-voice traffic-class local-voice**	Specifies that routed voice traffic will be processed like local generated voice traffic.
9	[*name*] **(pf-srvp)**[<name-in>]**# source traffic-class local-voice**	Enters traffic-class configuration mode
10	[*name*] **(src)**[local-v~]**# priority**	Specifies that local-voice has priority. Because route-voice is mapped to local-voice, also routed-voice has priority.
11	[*name*] **(src)**[local-v~]**# context ip**	Changes to IP configuration mode
12	[*name*] **(ctx-ip)**[router]**# interface** <if-wan>	Enters WAN interface configuration mode
13	[*name*] **(if-ip)**[<if-wan>]**# use profile service-policy** <name-in> **in**	Assigns the downlink profile on the WAN interface.
14	[*name*] **(if-ip)**[<if-wan>]**# use profile service-policy** <name-out> **out**	Assigns the uplink profile on the WAN interface.

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 IPLink software a service policy profile is used on a certain IP interface, as shown in figure 28.

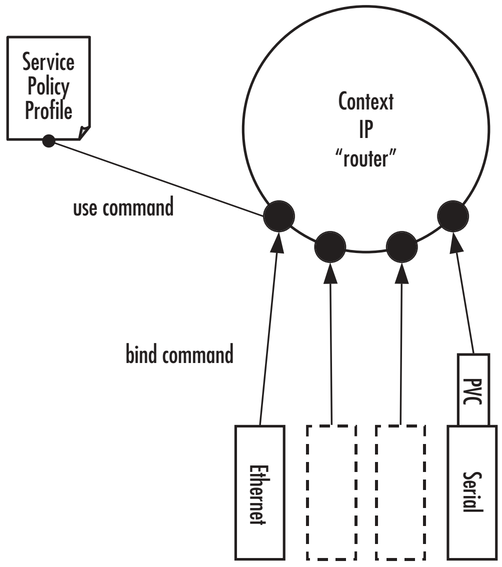


Figure 28. Using a Service Policy Profile on an IP Interface

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

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

Mode: Interface

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

```
IPLink>enable
IPLink#configure
IPLink(cfg)#context ip router
IPLink(ctx-ip)[router]#interface wan
IPLink(if-ip)[wan]#use profile service-policy Voice_Prio out
```

Displaying link arbitration status

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

Mode: Operator execution

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

Example: Displaying link arbitration status

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

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

Displaying link scheduling profile information

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

Mode: Administrator execution

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

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

Enable statistics gathering

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

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

Table 10. Values defining detail of the queuing statistics

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

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

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

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 4, defines the verbosity of the command output.

Example: Enable statistics gathering for all queues of a profile

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

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

Chapter 15 **Serial port configuration**

Chapter contents

Introduction

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

- Serial port configuration task list
- Configuration tasks
- Examples

The IPLink 2835 and 2821 Series support V.35 and X.21 serial interfaces. In this chapter, both products will be collectively referred to as *IPLink devices*. 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 IPLink software for specific information regarding the connector pinout and the selection of cables to connect with third-party equipment.

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

Serial port configuration task list

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

- Disabling an interface (see page 169)
- Enabling an interface (see page 169)
- Configuring the serial encapsulation type (see page 170)
- Configuring the active clock edge (see page 171)
- Entering Frame Relay mode (see page 173)
- Configuring the LMI type (see page 173)
- Configuring the keep-alive interval (see page 174)
- Enabling fragmentation (see page 174)
- Entering Frame Relay PVC configuration mode (see page 176)
- Configuring the PVC encapsulation type (see page 177)
- Binding the Frame Relay PVC to IP interface (see page 177)
- Disabling a Frame Relay PVC (see page 179)
- Displaying Frame Relay information (see page 181)

Disabling an interface

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

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

This procedure describes how to shut down a serial interface

Mode: Administrator execution

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

Example: Disabling an interface

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

```
IPLink(cfg)#port serial 0 0
IPLink(prt-ser)[0/0]#shutdown
IPLink(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 IPLink, use the **no shutdown** command to enable the serial interfaces again. When you enable an interface, it has the state OPENED in the **show port serial** command display.

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

This procedure describes how to enable a serial interface.

Mode: Administrator execution

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

Example: Enabling an interface

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

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

Serial Interface Configuration
------------------------------

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

Configuring the serial encapsulation type

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

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

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

Mode: Administrator execution

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

Example: Configuring the serial encapsulation type

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

```
IPLink(cfg)#port serial 0 0
IPLink(prt-ser)[0/0]#encapsulation framerelay
IPLink(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 active clock edge

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

This procedure describes 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**	Configures 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**	Configures the serial interface to transmit on the negative edge of the clock (inverted).

Example: Configuring the active clock edge

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

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

Configuring the baudrate

A DCE interface has to provide the signal clocks. The X.21 DCE interface can provide different baudrates on its interface. The desired baudrate can be configured.

Note The actual baudrate may differ from the baudrate you configured.

This procedure describes how to set the baudrate for the serial interface.

Mode: Port serial

Step	Command	Purpose
1	***node***(**prt-ser**)[*slot/port*]# **baudrate** *baudrate*	Configures the baudrate for the serial interface.

Example: Configuring baudrate to 64,000 bps

The following example configures a baudrate of 64,000 bps on the serial interface. Verify that the command **show port serial detail 5** output displays the correct baudrate. *True baudrate* in the *Status* section shows the baudrate of the selected hardware.

```
IPLink(cfg)#port serial 0 0
IPLink(prt-ser)[0/0]#transmit-data-on-edge negative
IPLink(prt-ser)[0/0]#show port serial detail 5

HDLC Driver: 0x8496b8
=====================

 Slot:                             0
 Number of Ports:                  1

 Port: serial 0 0 0
 ------------------

  State:                           OPENED

  Configuration
   Hardware Port:                  X.21
   Port Type:                      DCE
   CRC:                            CRC-16
   Transmit Edge:                  Normal
   Max Frame Length:               1920
   Baudrate:                       64000 bps
   Recv Threshold:                 1

  Statistics
   Received frames:                116101
   Rx good frames:                 116099
   Rx CD lost:                     0
   Rx Overrun:                     0
   Rx CRC errors:                  0
```

```
        Rx abort sequence:                    0
        Rx non octet:                         2
        Rx frame len violation:               0
        Rx DPLL error:                        0
        Sent frames:                          116106
        Tx good frames:                       116106
        Tx CTS lost:                          0
        Tx underrun:                          0

       Status
        Link:                                 Up
        Control Line:                         enabled
        True Baudrate:                        64000 bps
```

Enter Frame Relay mode

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

This procedure describes how to enter the Frame Relay configuration mode

Mode: Administrator execution

Step	Command	Purpose
1	***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*]**#**	Displays the Frame Relay configuration mode prompt

Example: Enter Frame Relay mode

The following example shows how to enter into the Frame Relay configuration mode for the serial interface on slot 0 and port 0 of an IPLink.

```
IPLink(cfg)#port serial 0 0
IPLink(prt-ser)[0/0]#framerelay
IPLink(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 IPLink device and the Frame Relay provider equipment. If the IPLink device is attached to a public data network (PDN), the LMI type must match the type used on the public network.

You can set one of the following three types of LMIs on the IPLink devices:

- **ansi** for ANSI T1.617 Annex D,
- **gof** for *Group of 4*, which is the default for Cisco LMI, and
- **itu** for ITU-T Q.933 Annex A.

This procedure describes how to set the LMI type.

Mode: Frame Relay

Step	Command	Purpose
1	***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.

```
IPLink(cfg)#port serial 0 0
IPLink(prt-ser)[0/0]#framerelay
IPLink(frm-rel)[0/0]#lmi-type ansi
```

Configuring the keep-alive interval

A keep-alive interval must be set to configure the LMI. By default, this interval is 10 seconds and, according to the LMI protocol, must be less than the corresponding interval on the switch. The keep-alive interval in seconds, which is represented by *number*, has to be in the range from 1 to 3600.

This procedure describes how to set the keep-alive interval

Mode: Frame Relay

Step	Command	Purpose
1	***node*(frm-rel)[*slot/port*]#keepalive** *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 an IPLink.

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

Enabling fragmentation

The IPLink software supports the FRF.12 interface and end-to-end fragmentation of large IP packets to reduce the delay imposed on time-sensitive packets on slow links (less than 512 kbps). As opposed to IP fragmentation (also supported by IPLink software) Frame Relay fragmentation is transparent to the IP layer. This leaves IP packets unchanged, which may be important for IP-based applications susceptible to IP fragmentation.

This procedure describes how to enable Frame Relay fragmentation

Mode: Frame Relay

Step	Command	Purpose
1	***node*(cfg)#port serial** *slot port*	Selects the serial interface on slot and port.
2	***node*(prt-ser)[*0/0*]#framerelay**	Enters Frame Relay configuration mode.
3	***node*(frm-rel)[*0/0*]#use profile service-policy** *name* **out**	Uses the previously defined service policy profile on Frame Relay layer (and not on IP interface level) in outward direction.
4	***node*(frm-rel)[*0/0*]#fragment** *size*	Defines the maximum size (in Bytes) of the Frame Relay payload (excluding Frame Relay header and trailer overhead) for all PVCs (FRF.12 interface fragmentation). See also the table below
5	***node*(frm-rel)[*0/0*]#pvc** *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*	Defines the maximum size (in bytes) of the Frame Relay payload (excluding Frame Relay header and trailer overhead) for this PVC only (FRF.12 end-to-end fragmentation). See also the table below

Note For proper functioning, do not specify a scheduler mode (burst-shaper, burst-WFQ, shaper, WFQ) for the Frame Relay service policy profile. Furthermore, do not use the Frame Relay service policy profile on the IP layer, but rather on the Frame Relay layer (mode *framerelay*). Make sure time-sensitive traffic is being given priority over data (command **source class local-voice priority**).

Note FRF.12 end-to-end fragmentation and FRF.12 interface fragmentation are incompatible. Thus make sure that both ends of a Frame Relay link run the same fragmentation mode.

Note When running data and voice over a Frame Relay link, it is advisable to only configure fragmentation for the PVC that carries data traffic. This way, fragmentation protocol overhead and fragmentation processing overhead is only spent for data traffic—voice packets (whose length should be smaller than the fragmentation length) do not consume processing power and protocol overhead for fragmentation.

The purpose of end-to-end FRF.12 fragmentation is to support real-time and non-real-time data packets on lower-speed links without causing excessive delay to the real-time data. The FRF.12 Implementation Agreement defines FRF.12 fragmentation. This standard was developed to allow long data frames to be fragmented into smaller pieces (fragments) and interleaved with real-time frames. In this way, real-time and non-real-time data frames can be carried together on lower-speed links without causing excessive delay to the real-time traffic. End-to-end FRF.12 fragmentation is recommended for use on permanent virtual circuits (PVCs) that share links with other PVCs transporting voice and on PVCs transporting Voice over IP (VoIP).

The fragmentation size depends on the available bandwidth, the chosen codec, and its packet length:

- The less bandwidth available per call, the smaller the fragment size has to be configured.
- The shorter the voice packets, the smaller the fragment size can be configured.
- The smaller the fragment size, the bigger the overhead for long data packets.

The following table shows the minimum fragment size depending on the configured codec and its packet length without fragmenting the voice packets:

Codec (bytes)	Packet Period (ms)	Minimum Fragment Size
G.729	10	52
G.729	20	62
G.729	30	72
G.723	30	66
G.723	60	90
G.723	90	114
G.711	10	122
G.711	20	202
G.711	30	282

Entering Frame Relay PVC configuration mode

The permanent virtual circuit (PVC) is a virtual circuit that is permanently established. PVCs save bandwidth associated with circuit establishment and tear down in situations where certain virtual circuits must exist all the time.

The Frame Relay network provides a number of virtual circuits that form the basis for connections between stations attached to the same Frame Relay network.

The resulting set of interconnected devices forms a private Frame Relay group, which may be either fully interconnected with a complete mesh of virtual circuits, or only partially interconnected.

In either case, each virtual circuit is uniquely identified at each Frame Relay interface by a Data Link Connection Identifier (DLCI). In most circumstances, DLCIs have strictly local significance at each Frame Relay interface.

Assigning a DLCI to a specified Frame Relay sub interface on the IPLink is done in the PVC configuration mode. The DLCI has to be in the range from 1 to 1022.

Note A maximum of eight PVCs can be defined.

This procedure describes how to enter the PVC configuration.

Mode: Frame Relay

Step	Command	Purpose
1	***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 an IPLink.

```
IPLink(cfg)#port serial 0 0
IPLink(prt-ser)[0/0]#framerelay
IPLink(frm-rel)[0/0]#pvc 1
IPLink(pvc)[1]#
```

Configuring the PVC encapsulation type

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

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

Mode: Frame Relay

Step	Command	Purpose
1	***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 an IPLink.

```
IPLink(cfg)#port serial 0 0
IPLink(prt-ser)[0/0]#framerelay
IPLink(frm-rel)[0/0]#pvc 1
IPLink(pvc)[1]#encapsulation rfc1490
```

Binding the Frame Relay PVC to IP interface

A newly created permanent virtual circuit (PVC) for Frame Relay has to be bound to an IP interface for further use. The logical IP interface has to be already defined and should be named according to the use of the serial

Frame Relay PVC. If serial Frame Relay PVC shall be used as WAN access, a suitable name for the logical IP interface could be *wan* as in figure 29 below.

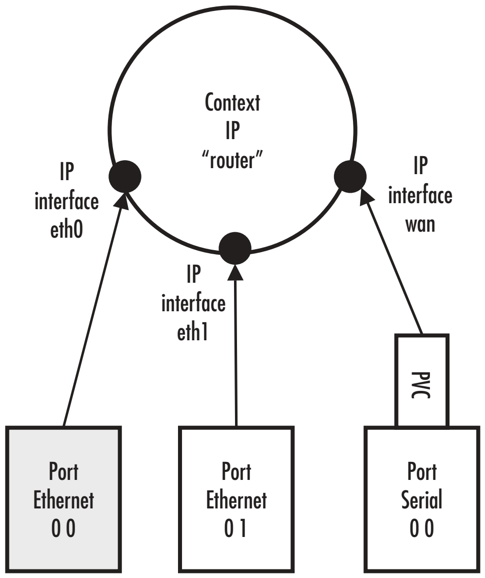


Figure 29. IP interface *wan* is bound to PVC 1 on port serial 0 0

This procedure describes how to bind the Frame Relay PVC DLCI on the serial interface to the logical IP interface *name*, which is related to the IP context router.

Mode: PVC

Step	Command	Purpose
1	***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 an IPLink.

```
IPLink(cfg)#port serial 0 0
IPLink(prt-ser)[0/0]#framerelay
IPLink(frm-rel)[0/0]#pvc 1
IPLink(pvc)[1]#bind interface wan router
```

Enabling a Frame Relay PVC

After binding Framerelay PVC to an ip interface it must be enabled for packet processing. This procedure activates the PVC by opening the bound ip interface.

This procedure describes how to enable Framerelay PVC for packet processing

Mode: PVC

Step	Command	Purpose
1	***node*(pvc)[dlci]#no shutdown**	Enables the Frame Relay PVC

Example: Disabling a Frame Relay PVC

The following example enables Frame Relay PVC with the DLCI 1 on the serial interface on slot 0 and port 0.

```
IPLink(cfg)#port serial 0 0
IPLink(prt-ser)[0/0]#framerelay
IPLink(frm-rel)[0/0]#pvc 1
IPLink(pvc)[1]#no shutdown
```

Check the PVC 1 status using **show running-config** and verify that the entry *no shutdown* occurs in the configuration part responsible for this PVC.

```
IPLink(pvc)[1]#show running-config
Running configuration:
#----------------------------------------------------------------#
#                                                                #
…
pvc 1
  encapsulation rfc1490
  bind interface wan router
  no shutdown
```

Disabling a Frame Relay PVC

Frame Relay PVCs can be disabled whenever it is necessary. Be aware that disabling a specific PVC also disables the related serial interface and vice versa.

This procedure describes how to disable the Frame Relay PVC DLCI on the serial interface.

Mode: PVC

Step	Command	Purpose
1	***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 an IPLink.

```
IPLink(cfg)#port serial 0 0
IPLink(prt-ser)[0/0]#framerelay
IPLink(frm-rel)[0/0]#pvc 1
IPLink(pvc)[1]#shutdown
```

Check the PVC 1 status by using **show running-config** and verify that the entry *shutdown* occurs in the configuration part responsible for this PVC.

```
IPLink(pvc)[1]#show running-config
Running configuration:
#-------------------------------------------------------------#
#                                                             #
#                                                         #
…
pvc 1
  encapsulation rfc1490
  bind interface wan router
  shutdown
  exit
…
```

Debugging Frame Relay

A set of commands is available to check the status of the Framerelay connections, fragmentation process and keepalive message exchange. Be aware that some monitors generate a lot of output and can seriously impact your system performance.This procedure describes how to display the Frame Relay configuration settings for the serial interface

Mode: Administrator execution

	Command	Purpose
	[no] debug framerelay	Prints the status of the different monitors (ON or OFF)
	[no] debug framerelay all	Enables/Disables all framerelay debug monitors
	[no] debug framerelay error	Enables/Disables monitor which prints only occurred errors.
	[no] debug framerelay lmi	Enables/Disables monitor which prints keepalive events and messages
	[no] debug framerelay management	Enables/Disables monitor which prints management and configuration events
	[no] debug framerelay packets	Enables/Disables monitor which prints dlci, size and fragmentation status of every incoming and outgoing packet. Be aware that this monitor can seriously impact your system performance.

Displaying serial port information

The following example shows the commands used to display serial port configuration settings.

```
HDLC Driver: 0x8496b8
=====================

 Slot:                           0
 Number of Ports:                1
HDLC Driver: 0x8496b8
=====================

 Slot:                           0
 Number of Ports:                1

 Port: serial 0 0 0
 ------------------

  State:                         OPENED

  Configuration
   Hardware Port:                X.21
   Port Type:                    DCE
   CRC:                          CRC-16
   Transmit Edge:                Normal
   Max Frame Length:             1920
   Baudrate:                     64000 bps
   Recv Threshold:               1
```

Displaying Frame Relay information

Since Frame Relay configuration for the serial interface is complex and requires many commands, it is helpful to list the frame relay configuration on screen.

This procedure describes how to display the Frame Relay configuration settings for the serial interface.

Mode: Port serial

Step	Command	Purpose
1	***node*(prt-ser)[*slot/port*]#show framerelay**	Displays Frame Relay information.

Example: Displaying Frame Relay information

The following example shows the commands used to display Frame Relay configuration settings.

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

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

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

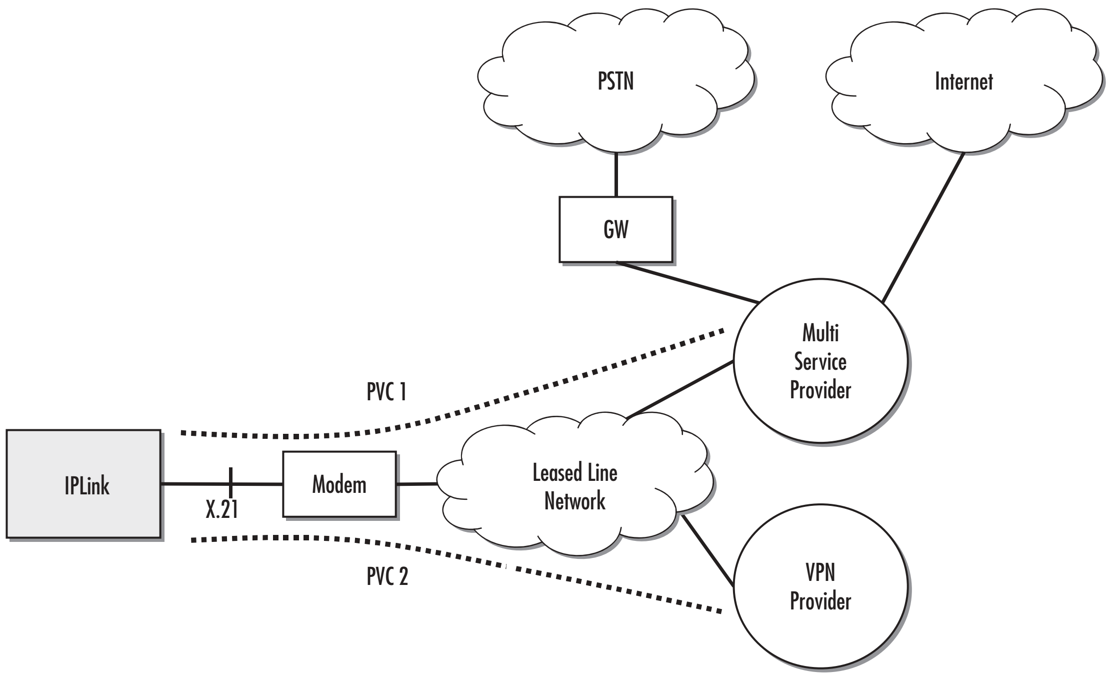


Figure 30. Typical Integrated Service Access Scenario with dedicated PVCs

Integrated service access

The example in figure 30 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 an IPLink.

The multi service provider (MSP) offers both Internet access and voice services based on IP. The virtual private network (VPN) provider offers secure interconnections of local access networks (LAN) via its public wide area network based on IP. Since both providers are working independently, the IPLink 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.

An IPLink is working as a DTE and accesses the leased line network via a leased line modem connected to the serial interface. The hardware port protocol X.21 is used on the serial interface on slot 0 and port 0.

Devices accessing the MSP and VPN services are attached to the 100 Mbps Ethernet port 0/0 on the IPLink. 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 30 has to be configured for the IPLink. 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 IPLink, 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.

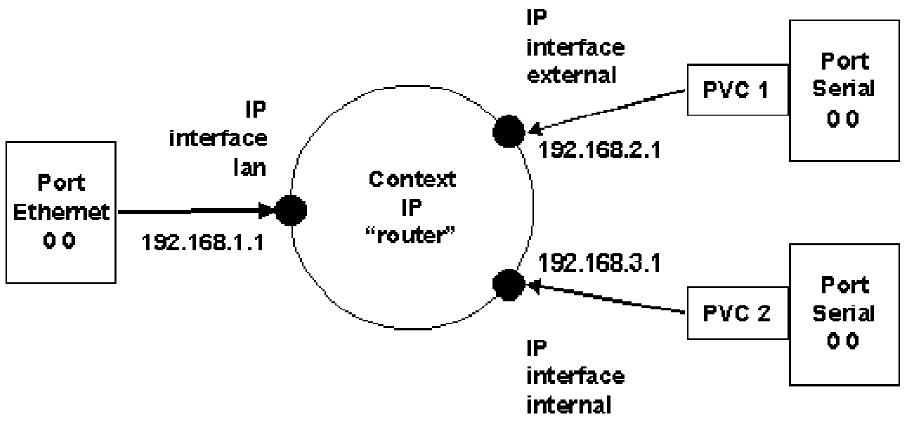


Figure 31. IP Context with logical IP interfaces bound to Ethernet port, serial port PVC 1 and PVC 2

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.

```
IPLink>enable
IPLink#configure
...
```

2. Set up the IP interface configuration first. Be aware that not all of the necessary settings are listed below.

```
IPLink(cfg)#context ip router
IPLink(ctx-ip)[router]#interface external
IPLink(if-ip)[external]#interface internal
IPLink(if-ip)[internal]#interface lan
IPLink(if-ip)[lan]#exit
IPLink(ctx-ip)[router]#interface internal
IPLink(if-ip)[internal]#ipaddress 192.168.3.1 255.255.255.0
IPLink(if-ip)[internal]#interface external
IPLink(if-ip)[external]#ipaddress 192.168.2.1 255.255.255.0
IPLink(if-ip)[external]#interface lan
IPLink(if-ip)[lan]#ipaddress 192.168.1.1 255.255.255.0
...
```

3. Define a voice profile which gives priority to voice packets. Set the rate limit according to the bandwidth available for voice and data on PVC 1 (512kBits/s in this case).

```
IPLink(cfg)#profile service-policy VoicePrio
IPLink(pf-srvpl)[VoicePr~]#rate-limit 512
IPLink(pf-srvpl)[VoicePr~]#source class local-voice
IPLink(src)[local-v~]#priority
IPLink(src)[local-v~]#source class local-default
IPLink(src)[local-d~]#priority
IPLink(src)[local-d~]#source class default
...
```

4. Configure the serial interface settings.

```
IPLink(cfg)#port serial 0 0
IPLink(prt-ser)[0/0]#shutdown
IPLink(prt-ser)[0/0]#encapsulation framerelay
IPLink(prt-ser)[0/0]#hardware-port x21
IPLink(prt-ser)[0/0]#port-type dte
...
```

5. Configure the Frame Relay. You must thus change to the Frame Relay configuration mode. Use the service-policy profile defined above to give voice priority over data.

```
IPLink(prt-ser)[0/0]#framerelay
IPLink(frm-rel)[0/0]#lmi-type ansi
IPLink(frm-rel)[0/0]#keepalive 20
IPLink(frm-rel)[0/0]#use profile service-policy VoicePrio out
...
```

6. Configure the introduced PVCs. Enable fragmentation for PVC 1. The voice uses codec G.723 at a packet size of 30ms, so the minimum fragment size must be 66 Bytes. Setting the fragment size to 300 (Bytes) introduces an additional delay of at most 4.7ms (300 * 8/512k) but does not cause too much fragmentation overhead on large data packets.

```
IPLink(frm-rel)[0/0]#pvc 1
IPLink(pvc)[1]#encapsulation rfc1490
IPLink(pvc)[1]#fragment 300
IPLink(pvc)[1]#bind interface external router
IPLink(pvc)[1]#no shutdown
IPLink(pvc)[1]#pvc 2
IPLink(pvc)[2]#encapsulation rfc1490
IPLink(pvc)[2]#bind interface internal router
IPLink(pvc)[2]#no shutdown
...
```

7. Check that the Frame Relay settings are correct.

```
IPLink(frm-rel)[0/0]#show framerelay

Framerelay Configuration:
Port            LMI-Type        Keepalive       Fragmentation
-------------------------------------------------------------
serial 0 0 0    ansi            20              disabled


PVC Configuration:
Port            DLCI     State     Fragment  Encaps    Binding
-----------------------------------------------------------------
serial 0 0 0    1        open      300       rfc1490   external@router
serial 0 0 0    2        open      disabled  rfc1490   internal@router
```

Chapter 16 T1/E1 port configuration

Chapter contents

Introduction

This chapter provides an overview of the T1/E1 ports, their characteristics and the tasks involved in the configuration.

The configurable parameters for the T1/E1 port are type (T1 or E1), clock mode (or source) (master or slave), line code (AMI, HDB3, or B8ZS), framing (CRC-4, ESF, or unframed), line-build-out (for T1 only) and encapsulation (channelized or HDLC).

A further feature is the creation and configuration of channel-groups.

T1/E1 port configuration task list

This section describes the configuration tasks for the T1/E1 port.

- Enable/Disable T1/E1 port
- Configuring the T1/E1 port type
- Configuring T1/E1 clock mode
- Configuring T1/E1 line code
- Configuring T1/E1 framing
- Configuring T1 line build out (LBO) (T1 only)
- Configuring E1 impedance/connector
- Configuring T1/E1 application mode
- Configuring T1/E1 LOS threshold
- Configuring T1 Loopback detection (T1 only)
- Configuring T1/E1 encapsulation
- Create a Channel-Group
- Configuring channel-group timeslots
- Configuring channel-group encapsulation
- Entering HDLC configuration mode
- Configuration HDLC CRC-type
- Configuring HDLC encapsulation

Enable/Disable T1/E1 port

By default, the T1/E1 port is disabled. The following command is used for enabling or disabling it.

Mode: port e1t1 <slot> <port>

Step	Command	Purpose
1	[*name*] **(prt-e1t1)**[slot/port]**# [no] shutdown**	Enable/Disable the T1/E1 port. Default: *shutdown* (which is disabled)

Configuring T1/E1 port-type

The T1/E1 Port can either work in T1 or in E1 (G.704) mode. This mode can be changed dynamically as long as no encapsulation or encapsulation 'hdlc' is set. Be aware that changing the port-type also resets the framing and linecode parameters to the default values of the new port-type. If port-type change is not allowed due to current configuration, an error message will be issued.

Mode: port e1t1 <slot> <port>

Step	Command	Purpose
1	[*name*] **(prt-e1t1)**[slot/port]**# port-type {e1 | t1}**	Changes operation mode of the port. Default: *e1*

Configuring T1/E1 clock-mode

The T1/E1 Port can either work in clock-master or in clock-slave mode. This setting defines the clock dependency of the internal data processing. In clock-master mode the internal data processing is running on an independent clock source. In clock-slave mode the clock source for internal data processing is recovered from the receive line interface. Be aware that always a port-pair of clock-master and clock-slave are connected together. In the other case the data transmission will fail due to bit failures.

Mode: port e1t1 <slot> <port>

Step	Command	Purpose
1	[*name*] **(prt-e1t1)**[slot/port]**# clock {master | slave}**	Configures the clock-mode of the port. Default: *master*

Configuring T1/E1 line-code

Three different line codes can be selected on the T1/E1 port whereas only 'ami' is standardized for E1 and T1. If the port is running in E1 mode, 'hdb3' is also configurable and in T1 mode 'b8zs'. If a linecode will be selected that is not standardized for the current port mode, an error message will be advised.

Mode: port e1t1 <slot> <port>

Step	Command	Purpose
1	[*name*] **(prt-e1t1)**[slot/port]**# linecode {ami | b8zs | hdb3}**	Configures the line-code of the port. Default for e1: *hdb3* Default for t1: *b8zs*

Configuring T1/E1 framing

Four framing formats are available for selection on the T1/E1 port. *Unframed* can only be used if the encapsulation is set for *hdlc*. All other currently available upper layer (encapsulation) protocols do not run in unframed mode, but in one of the framed modes.

In structured mode, E1 can be configured for *crc4* or *non-crc4*. T1 has a single framed option, *esf*.

The advantage of the unframed mode (obviously with *hdlc* encapsulation) is the utilization of the whole link speed for user data transmission, 2.048MBit/s for E1 and 1.544MBit/s for T1. However note that HDLC has its own overhead which decreases the actual data rate.

Mode: port e1t1 <slot> <port>

Step	Command	Purpose
1	[*name*] **(prt-e1t1)**[slot/port]**# framing {crc4 | non-crc4 | esf | unframed}**	Configures the framing of the port. E1 mode formats are: *crc4, non-crc4, unframed.* T1 mode formats are: *esf, unframed.* Default for **e1**: *crc4* Default for **t1**: *esf*

Configuring T1/E1 line-build-out (T1 only)

The line build out configuration is used in long haul applications to prevent cross talk in the far end device.

Mode: port e1t1 <slot> <port>

Step	Command	Purpose
1	[*name*] **(prt-e1t1)**[slot/port]**# line-build-out {0 | -7.5 | -15 | -22.5}**	Specifies the pulse attenuation in dB on the line interface. Default for t1: *0* dB

Configuring T1/E1 used-connector (E1 only)

If the T1/E1 WAN-Card provides several line interface connector types the command specifies which one is currently in use. Sure, the signal is always on all connectors available but dependent on the wiring technology the internal impedance matching must be adapted (RJ45 = 120 Ohm; BNC = 75 Ohm).

Mode: port e1t1 <slot> <port>

Step	Command	Purpose
1	[*name*] **(prt-e1t1)**[slot/port]**# used-connector {bnc | rj45}**	Specifies the currently used connector. Default for e1: *rj45*

Configuring T1/E1 application mode

The T1/E1 port can be configured to work in either short-haul or in long-haul mode. **Short-haul** is the default application and should be used for transmission distances up to **180m/600ft**. For transmission distances up to **1800m/6000ft**, select the **long-haul** application.

Mode: port e1t1 <slot> <port>

Step	Command	Purpose
1	[*name*] **(prt-e1t1)**[*slot/port*]**#application {long-haul | short-haul}**	Specifies the e1/t1 application mode Default: *short-haul*

Configuring T1/E1 LOS threshold

This command takes effect only if the T1/E1 port is configured for **long-haul** applications. It specifies the sensitivity for **Loss Of Signal threshold**. A signal suffers more attenuation over long distances than over short distances. Therefore the LOS-Threshold must be set higher for longer transmission distances. This command has a default value of -46dB what should be enough for distances up to 1600 m/5250 ft.

Mode: port e1t1 <slot> <port>

Step	Command	Purpose
1	[*name*] **(prt-e1t1)**[*slot/port*]**#los-threshold {-4dB | -6dB | -8dB ... -46dB | -48dB}**	Specifies Loss Of Signal Threshold Default: *-46dB*

Configuring T1 Loopback detection

In T1 mode the E1/T1 interface module has the capability for autodetection of inband sent loopback codes. Once a loopback-up code is detected, the module automatically enables the proper loopback function and disables it a soon as the corresponding loopback-down code appears. This feature is used by carrier equipment for testing the line to the customer. It sends the loopback-up code to the customer device, then subsequently starts, for example, a Pseudo Random Bit Sequence (PRBS) to determinate the quality of the connection..

Depending on the configured T1 framing, the right loopback code detection mode will be enabled as soon as the command `loop-back auto-detection` will be executed. For framing type uses a different loopback code detection mechanism:

- **ESF**: The loopback codes are transmitted via the 4kBit/s EOC-Channel, that is part of the 8kBit/s F-Bit Channel. The following codes are supported:

Command	Binary Code
Line Loopback Activate	`0 000111 0`
Line Loopback Deactivate	`0 011100 0`
Payload Loopback Activate	`0 001010 0`
Payload Loopback Deactivate	`0 011001 0`
Universal Loopback Deactivate	`0 010010 0`
Loopback Retention	`0 010101 0`

- **SF** and **Unframed**: An inband loop code pattern is sent for at least 5 seconds in all 24 timeslots. The following codes are supported:

Command	Binary RepetitionCode
Line Loopback Activate	`00001`
Line Loopback Deactivate	`001`

The command has three other options that allow you to manually switch on/off different loops. All these additional options are applicable in T1 and E1 mode.

The 'line-interface' loop sends back the whole link bandwidth (2048kBit/s or 1544kBit/s).

In 'payload' the entire user data bandwidth (1984 kbps or 1536 kbps) is looped back.

For some tests it is helpful to loop back the system data. For example, system data are sent from the router to the T1/E1 port. To switch on this feature the option 'back-plane' must be selected.

Mode: port e1t1 <slot> <port>

Step	Command	Purpose
1	**[** *name***] (prt-e1t1)[***slot/port***]#[no] loop-back {line-interface | back-plane | payload | auto-detection}**	Enables/Disables type of data loopback, line-interface, payload, back-plane, or auto-detection. Default: disabled

Configuring T1/E1 encapsulation

In release R3.20 only 'hdlc' encapsulation is available on a T1/E1 port. Once encapsulation is configured as '**hdlc**,' the 'hdlc' submode can be entered for selecting the next encapsulation type like **ppp** or framerelay. Dependent on the port-type, the encapsulation 'hdlc' selects automatically all timeslots of the port for data transmission (**1-31** for e1 and **1-24** for t1).

It is also possible to use the port in **channelized** mode. In "channelized" mode, the user selects less than the total number of timeslots for the channel (1-31 for E1, 1 – 24 for T1) is able to configure single or multiple timeslots for data transmission. To use this feature the encapsulation must be configured for 'channelized;' afterwards the **channel-group** command is used to create the channel-group. In the channel-group configuration mode, the user selects the specific timeslots, and the encapsulation 'hdlc' will be available again. Once the encapsulation of a T1/E1 port is set to 'channelized' it is not possible to change the port-type again or to use the 'unframed' framing format.

Mode: port e1t1 <slot> <port>

Step	Command	Purpose
1	**[***name***] (prt-e1t1)[**slot/port**]#[no] encapsulation {channelized | hdlc}**	Specifies the encapsulation type of the T1/E1 port. Default: *no encapsulation*

Create a Channel-Group

If the desired encapsulated channel uses only selected time slots (not the entire T1/E1), then it is necessary to set up a channel-group. To create a channel-group, set the T1/E1 port's encapsulation to *channelized*. (See section "In T1 mode the E1/T1 interface module has the capability for autodetection of inband sent loopback codes. Once a loopback-up code is detected, the module automatically enables the proper loopback function and disables it a soon as the corresponding loopback-down code appears. This feature is used by carrier equipment for testing the line to the customer. It sends the loopback-up code to the customer device, then subsequently starts, for example, a Pseudo Random Bit Sequence (PRBS) to determinate the quality of the connection..".) On creating a new channel-group the channel-group configuration mode is immediately entered. To remove an existing channel-group the 'no' form of the command has to be used.

Mode: port e1t1 <slot> <port>

Step	Command	Purpose
1	[*name*] **(prt-e1t1)**[slot/port]**#[no] channel-group group-name**	Enters the channel-group configuration mode of *group-name*. If the group does not yet exist a new one will be created. The 'no' form of the command removes an existing channel-group.

Configuring Channel-Group Timeslots

The 'timeslots' command configures an arbitrary sequence of timeslots for use in data transmission. The syntax of the command accepts comma-separated groups of timeslots. A group can be a single timeslot or a range of timeslots. The channel-group timeslots do not have to be contiguous. The 'no' form of the command releases all previously selected timeslots.

Example:

```
>timeslots 1,4,6            Selects three timeslots (1, 4 an 6)
>timeslots 1,4-6            Selects four timeslots (1, 4, 5 and 6)
>timeslots 1-3,4-6          Selects six timeslots (1, 2, 3, 4, 5 and 6)
```

Mode: channel-group *group-name*

Step	Command	Purpose
1	[name] **(ch-grp)**[group-name]**#[no] timeslots** *timeslots*	Selects the timeslots to be used. Default: *no timeslots*

Configuring Channel-Group Encapsulation

In the channel-group configuration mode only the encapsulation type 'hdlc' is available. For more details see 'Configuring T1/E1 Encapsulation'.

Mode: channel-group *group-name*

Step	Command	Purpose
1	[name] **(ch-grp)**[group-name]**#[no] encapsulation hdlc**	Specifies the encapsulation type of the channel-group. Default: *no encapsulation*

Entering HDLC Configuration Mode

The hdlc configuration mode can be entered either from the "port T1/E1" configuration mode or from the "channel-group" configuration mode. If you cannot enter the hdlc mode, it may be due to an invalid or incomplete configuration, and an error message will be issued. In "port T1/E1" configuration mode, you only need to set the encapsulation for 'hdlc' in order to enter the hdlc configuration mode. In "channel-group" configuration mode the encapsulation must be set to 'hdlc' as well followed by configuring at least one timeslot per the 'timeslots' command.

Mode: port e1t1 <slot> <port>

Step	Command	Purpose
1	[*name*] **(prt-e1t1)**[slot/port]**# hdlc**	Entering the hdlc configuration mode

Mode: channel-group <group>

Step	Command	Purpose
1	[name] **(ch-grp)**[group-name]**#hdlc**	Entering the hdlc configuration mode

Configuring HDLC CRC-Type

This command specifies the length of the checksum for calculating the CRC of the hdlc-frame. It can be either a 16-bit or a 32-bit checksum.

Mode: hdlc

Step	Command	Purpose
1	[*name*] **(hdlc)#crc-type {crc16 | crc32}**	Selects the checksum-type to be used. Default: *crc16*

Configuring HDLC Encapsulation

The hdlc encapsulation command specifies what kinds of upper layer data are contained in the hdlc frames. Two encapsulation types are available, framerelay and ppp. Once the hdlc configuration mode has been entered, the procedure for setting up framerelay or ppp is exactly the same as for an X.21/V.35 serial port. For that reason, see chapter 15, "Serial port configuration" on page 167 for more details about frame relay configuration and chapter 25, "PPP configuration" on page 270 for PPP configuration.

Mode: hdlc

Step	Command	Purpose
1	[*name*] **(hdlc)#encapsulation {framerelay | ppp}**	Specifies the encapsulation type of hdlc. Default: *no encapsulation*

T1/E1 Configuration Examples

Here is a group of four configuration examples.

- Example 1: Frame Relay without a channel-group
- Example 2: Frame Relay with a channel-group
- Example 3: PPP without a channel-group
- Example 4: PPP with a channel-group

Example 1: Frame Relay without a channel-group

```
port e1t1 0 0
  port-type e1
  framing crc4
  encapsulation hdlc

  hdlc
    encapsulation framerelay

    framerelay
      lmi-type itu

      pvc 100
        encapsulation rfc1490
        bind interface pvc100 router
        no shutdown

port e1t1 0 0
  no shutdown
```

Example 2: Framerelay with a channel-group

```
port e1t1 0 0
  port-type e1
  framing crc4
  encapsulation channelized

  channel-group myGroup
    timeslots 13-17
    encapsulation hdlc

    hdlc
      encapsulation framerelay

      framerelay
        lmi-type itu

        pvc 100
          encapsulation rfc1490
          bind interface pvc100 router
          no shutdown

port e1t1 0 0
  no shutdown
```

Example 3: PPP without a channel-group

```
port e1t1 0 0
  port-type e1
  framing crc4
  encapsulation hdlc

  hdlc
    encapsulation ppp
    bind interface myPPP router

port e1t1 0 0
  no shutdown
```

Example 4: PPP with a channel-group

```
port e1t1 0 0
  port-type e1
  framing crc4
  encapsulation channelized

  channel-group yourGroup
    timeslots 1,9,16,23
    encapsulation hdlc

    hdlc
      encapsulation ppp
      bind interface myPPP router

port e1t1 0 0
  no shutdown
```

Chapter 17 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 IPLink software.

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 IPLink software 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 IPLink receives an incoming packet, it checks the destination address, and attempts to associate this address with a next-hop address and outgoing interface. Routing algorithms must converge rapidly — i.e. all routers must agree on optimal routes. When a network event causes routes either to go down or to become unavailable, routers distribute routing update messages that permeate networks, causing recalculation of optimal routes that are eventually agreed upon by all routers. Routing algorithms that converge slowly can cause routing loops or network outages. Many algorithms can quickly select next-best paths and adapt to changes in network topology.

Static routing

Static routing involves packet forwarding on the basis of static routes configured by the system administrator. Static routes work well in environments where network traffic is relatively predictable and where the network topology is relatively simple. In contrast, dynamic routing algorithms adjust to changing network circumstances by analyzing incoming routing update messages. RIP uses dynamic routing algorithms.

Basic IP routing configuration task list

To configure IP routes, perform the tasks described in the following sections. The tasks in the first two sections are required; the task in the remaining section is optional, but might be required for your application.

- Configuring static IP routes
- Deleting static IP routes (see page 197)
- Displaying IP route information (see page 198)

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 the device at 172.17.100.2. The Ethernet port 0 1 has the address 172.17.100.1/24 and is bound to the interface *wan*.

```
IPLink>enable
IPLink#configure
IPLink(cfg)#context ip router
IPLink(ctx-ip)[router]#route 20.0.0.0 255.255.255.0 172.17.100.2
```

The route is added to the routing database with the default metric 0. The router will forward packets to the 20.0.0.0 network via the interface *wan* to the router on 172.17.100.2.

Deleting static IP routes

The **no** form of the **route** command deletes a static IP route from the routing table.

This procedure describes how to delete one or more static IP routes from the routing table

Mode: Administrator execution

Step	Command	Purpose
1	***node*(cfg)#context ip router**	Enters the IP router context
2	***node*(ctx-ip)[router]#no route** *network mask* **{***address* **|** *interface***}**	Deletes a static route

Example: Deleting a static IP route

In the following example, the route for packets to network 20.0.0.0/24, which are routed to device with IP address 172.17.100.2, shall be deleted.

```
IPLink>enable
IPLink#configure
IPLink(cfg)#context ip router
IPLink(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.

```
IPLink>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 32 shows an Internetwork consisting of three routers, an IPLink 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 IPLink 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*.

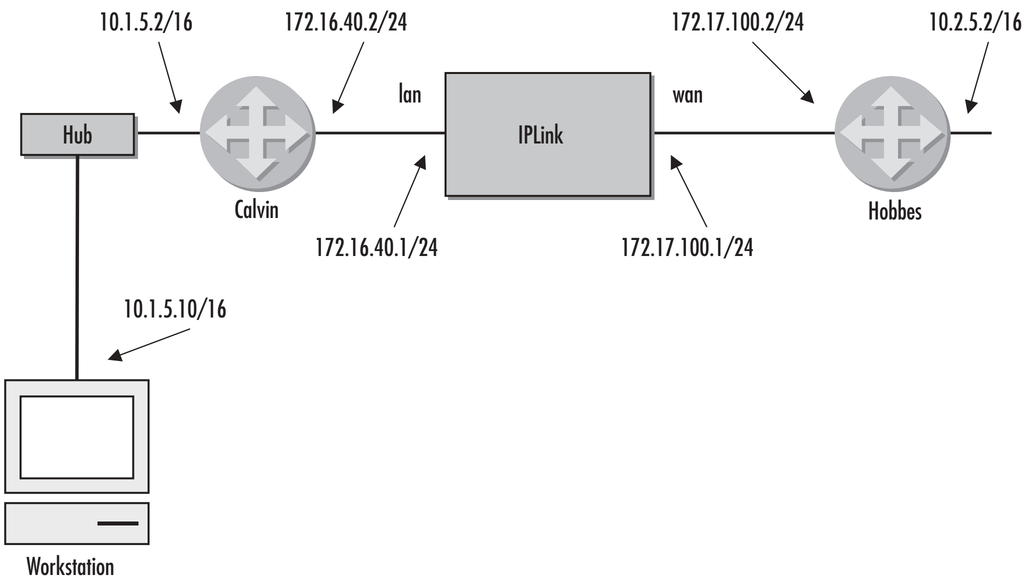


Figure 32. Internetwork with three routers and four networks

The necessary routing-table entries for the scenario described are listed below.

```
IPLink>enable
IPLink#configure
IPLink(cfg)#context ip router
IPLink(ctx-ip)[router]# route 10.1.5.10 255.255.255.255 172.16.40.2
IPLink(ctx-ip)[router]# route 10.2.0.0 255.255.0.0 172.17.100.2
IPLink>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
```

Changing the default UDP port range for RTP and RTCP

The UDP port range to be used for RTP streams can be configured using the following procedure:

Mode: context ip

Step	Command	Purpose
1	[*name*] **(ctx-ip)**[router]**# rtp-port-range** <start-port> <end-port>	Define the UDP port range for RTP/RTCP streams.

Chapter 18 **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 IPLink software, it includes the following sections:

- Routing protocol
- RIP configuration task list (see page 203)

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 IPLink software 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

An IPLink 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.

IPLink software 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 204)
- Specifying the send RIP version (see page 204)
- Specifying the receive RIP version (see page 205)
- Enabling RIP learning (see page 205)
- Enabling an interface to receive RIP (see page 206)
- Enabling RIP announcing (see page 206)
- Enabling RIP auto summarization (see page 207)
- Specifying the default route metric (see page 207)
- Enabling RIP split-horizon processing (see page 208)
- Enabling the poison reverse algorithm (see page 208)
- Enabling holding down aged routes (see page 209)
- Displaying RIP Configuration of an IP interface (see page 209)
- Displaying global RIP information (see page 210)

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 an IPLink.

```
IPLink(cfg)#context ip router
IPLink(ctx-ip)[router]#interface wan
IPLink(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 an IPLink.

```
IPLink(cfg)#context ip router
IPLink(ctx-ip)[router]#interface wan
IPLink(if-ip)[wan]#rip receive
```

Specifying the send RIP version

By default, IPLink software application software sends RIP 1compatible packets. The IPLink software 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 an IPLink.

```
IPLink(cfg)#context ip router
IPLink(ctx-ip)[router]#interface wan
IPLink(if-ip)[wan]#rip send version 1compatible
```

Specifying the receive RIP version

By default, IPLink software application software receives RIP version 1 and version 2 packets. IPLink software 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 an IPLink.

```
IPLink(cfg)#context ip router
IPLink(ctx-ip)[router]#interface wan
IPLink(if-ip)[wan]#rip receive version 1or2
```

Enabling RIP learning

A new route is added to the local routing table, if the routing update contains a route to a destination that does not already exist. If the update describes a route whose destination is already in the local table, the new route is used only if it has a lower cost. The cost of a route is determined by adding the cost of reaching the gateway that sent the update to the metric contained in the RIP update packet. If the total metric is less than the metric of the current route, the new route is used. IPLink software 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 an IPLink.

```
IPLink(cfg)#context ip router
IPLink(ctx-ip)[router]#interface wan
IPLink(if-ip)[wan]#rip learn host
IPLink(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 an IPLink device.

```
IPLink(cfg)#context ip router
IPLink(ctx-ip)[router]#interface lan
IPLink(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 IPLink devices 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 an IPLink.

```
IPLink(cfg)#context ip router
IPLink(ctx-ip)[router]#interface wan
IPLink(if-ip)[wan]#rip announce default
IPLink(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 IPLink 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 an IPLink.

```
IPLink(cfg)#context ip router
IPLink(ctx-ip)[router]#interface wan
IPLink(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 an IPLink receives a routing update that contains a new or changed destination-network entry, the IPLink 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 an IPLink receives a routing update that contains a new or changed entry, and if increasing the metric value by one causes the metric to be infinity (i.e. 16), the network destination is considered unreachable.

Because metrics cannot be directly compared, you must specify the default metric in order to designate the cost of the redistributed route used in RIP updates. All routes that are redistributed will use the default metric.

Setting the default route metric, which is a number, indicating the distance to the destination network element, e.g. another router or IPLink 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 an IPLink.

```
IPLink(cfg)#context ip router
IPLink(ctx-ip)[router]#interface wan
IPLink(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 an IPLink.

```
IPLink(cfg)#context ip router
IPLink(ctx-ip)[router]#interface wan
IPLink(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 an IPLink.

```
IPLink(cfg)#context ip router
IPLink(ctx-ip)[router]#interface wan
IPLink(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 an IPLink.

```
IPLink(cfg)#context ip router
IPLink(ctx-ip)[router]#interface wan
IPLink(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 an IPLink.

```
IPLink(cfg)#context ip router
IPLink(ctx-ip)[router]#interface wan
IPLink(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

IPLink software 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 an IPLink.

```
IPLink(cfg)#show rip
RIP information:
rip enabled
```

Chapter 19 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 IPLink software.

This chapter includes the following sections:

- About access control lists
- Access control list configuration task list (see page 214)
- Examples (see page 224)

About access control lists

This section briefly describes what access lists do, why and when you should configure access lists, and basic versus advanced access lists.

What access lists do

Access lists filter network traffic by controlling whether routed packets are forwarded, dropped or blocked at the router's interfaces. Your router examines each packet to determine whether to forward or drop the packet, based on the criteria you specified within the access lists.

Access list criteria could be the source address of the traffic, the destination address of the traffic, the upper-layer protocol, or other information.

Note Sophisticated users can sometimes successfully evade or fool basic access lists because no authentication is required.

Why you should configure access lists

There are many reasons to configure access lists. For example, you can use access lists to restrict contents of routing updates, or to provide traffic flow control. But one of the most important reasons to configure access lists is to provide security for your network, and this is the reason explored in this chapter.

You should use access lists to provide a basic level of security for accessing your network. If you do not configure access lists on your router, all packets passing through the router could be allowed onto all parts of your network.

For example, access lists can allow one host to access a part of your network, and prevent another host from accessing the same area. In figure 33 host A is allowed to access the Human Resources network and host B is prevented from accessing the Human Resources network.

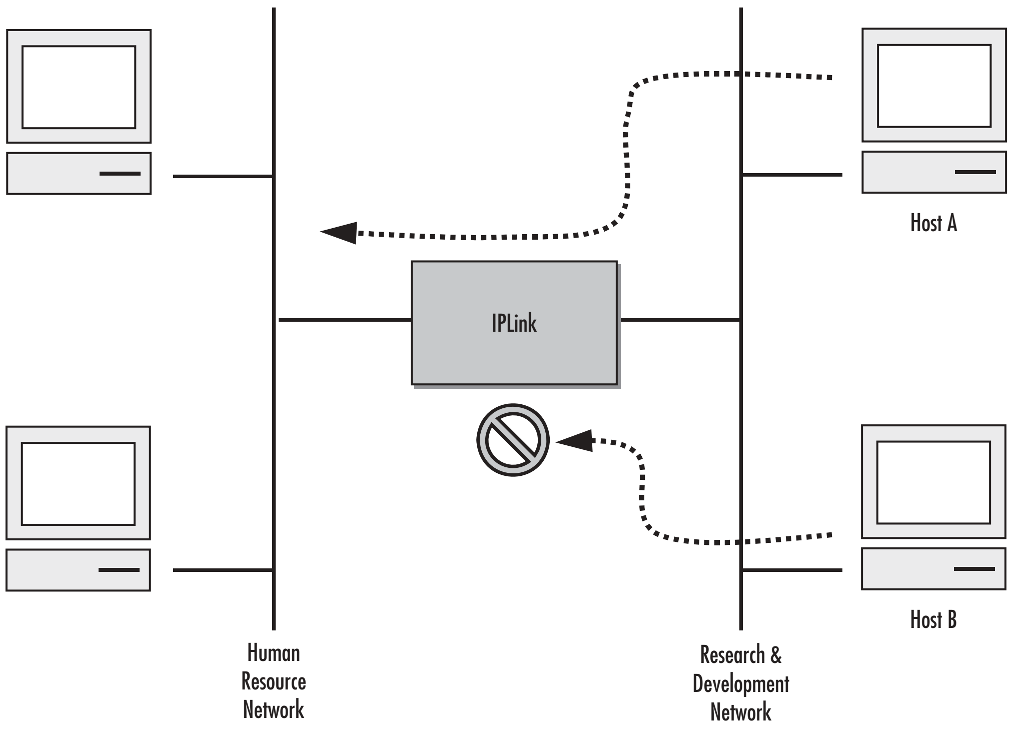


Figure 33. Using traffic filters to prevent traffic from being routed to a network

You can also use access lists to decide which types of traffic are forwarded or blocked at the router interfaces. For example, you can permit e-mail traffic to be routed but at the same time block all Telnet traffic.

When to configure access lists

Access lists should be used in firewall routers, which are often positioned between your internal network and an external network such as the Internet. You can also use access lists on a router positioned between two parts of your network, to control traffic entering or exiting a specific part of your internal network.

To provide the security benefits of access lists, you should configure access lists at least on border routers, i.e. those routers situated at the edges of your networks. This provides a basic buffer from the outside network or from a less controlled area of your own network into a more sensitive area of your network.

On these routers, you should configure access lists for each network protocol configured on the router interfaces. You can configure access lists so that inbound traffic or outbound traffic or both are filtered on an interface.

Features of access control lists

The following features apply to all IP access control lists:

- A list may contain multiple entries. The order access of control list entries is significant. Each entry is processed in the order it appears in the configuration file. As soon as an entry matches, the corresponding action is taken and no further processing takes place.

- All access control lists have an implicit *deny ip any any* at the end. A packet that does not match the criteria of the first statement is subjected to the criteria of the second statement and so on until the end of the access control list is reached, at which point the packet is dropped.
- Filter types include IP, Internet Control Message Protocol (ICMP), Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and Stream Control Transmission Protocol (SCTP).
- An empty access control list is treated as an implicit *deny ip any any* list.

Note Two or more administrators should not simultaneously edit the configuration file. This is especially the case with access lists. Doing this can have unpredictable results.

Once in access control list configuration mode, each command creates a statement in the access control list. When the access control list is applied, the action performed by each statement is one of the following:

- **permit** statement causes any packet matching the criteria to be accepted.
- **deny** statement causes any packet matching the criteria to be dropped.

To delete an entire access control list, enter configuration mode and use the **no** form of the **profile acl** command, naming the access list to be deleted, e.g. no profile acl *name*. To unbind an access list from the interface to which it was applied, enter the IP interface mode and use the **no** form of the access control list command.

Access control list configuration task list

To configure an IP access control list, perform the tasks in the following sections.

- Mapping out the goals of the access control list
- Creating an access control list profile and enter configuration mode (see page 215)
- Adding a filter rule to the current access control list profile (see page 215)
- Adding an ICMP filter rule to the current access control list profile (see page 217)
- Adding a TCP, UDP or SCTP filter rule to the current access control list profile (see page 219)
- Binding and unbinding an access control list profile to an IP interface (see page 221)
- Displaying an access control list profile (see page 222)
- Debugging an access control list profile (see page 222)

Mapping out the goals of the access control list

To create an access control list you must:

- Specify the protocol to be filtered
- Assign a unique name to the access list
- Define packet-filtering criteria

A single access control list can have multiple filtering criteria statements.

Before you begin to enter the commands that create and configure the IP access control list, be sure that you are clear about what you want to achieve with the list. Consider whether it is better to deny specific accesses and permit all others or to permit specific accesses and deny all others.

Note Since a single access control list can have multiple filtering criteria statements, but editing those entries online can be tedious. Therefore, we recommend editing complex access control lists offline within a configuration file and downloading the configuration file later via TFTP to your IPLink device.

Creating an access control list profile and enter configuration mode

This procedure describes how to create an IP access control list and enter access control list configuration mode

Mode: Administrator execution

Step	Command	Purpose
1	***node*(cfg)#profile acl** *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.

```
IPLink>enable
IPLink#configure
IPLink(cfg)#profile acl WanRx
IPLink(pf-acl)[WanRx]#
```

Adding a filter rule to the current access control list profile

The commands **permit** or deny are used to define an IP filter rule. This procedure describes how to create an IP access control list entry that permits access

Mode: Profile access control list

Step	Command	Purpose
1	***node*(pf-acl)[*name*]#permit ip** {*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 14, "Link scheduler configuration" on page 148.
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.

```
IPLink(cfg)#profile acl WanRx
IPLink(pf-acl)[WanRx]#permit ip host 62.1.2.3 host 193.14.2.11 cos Urgent
IPLink(pf-acl)[WanRx]#permit ip 62.1.2.3 0.0.255.255 host 193.14.2.11
IPLink(pf-acl)[WanRx]#permit ip 97.123.111.0 0.0.0.255 host 193.14.2.11
IPLink(pf-acl)[WanRx]#deny ip any any
IPLink(pf-acl)[WanRx]#exit
IPLink(cfg)#
```

Adding an ICMP filter rule to the current access control list profile

The command **permit** or **deny** are used to define an ICMP filter rule. Each ICMP filter rule represents an ICMP access of control list entry.

This procedure describes how to create an ICMP access control list entry that permits access

Mode: Profile access control list

Step	Command	Purpose
1	***node*(pf-acl)**[*name*]**#permit icmp** {*src src-wildcard* **| any | host** *src*} {*dest dest-wildcard* **| any | host** *dest*} **[msg** *name* **| type** *type* **| type** *type* **code** *code*] **[cos** *group*]	Creates an ICMP access of control list entry that permits access defined according to the command options

This procedure describes how to create an ICMP access control list entry that denies access

Mode: Profile access control list

Step	Command	Purpose
1	***node*(pf-acl)**[*name*]**#deny icmp** {*src src-wildcard* **| any | host** *src*} {*dest dest-wildcard* **| any | host** *dest*} **[msg** *name* **| type** *type* **| type** *type* **code** *code*] **[cos** *group*]	Creates an ICMP access of control list entry that denies access defined according to the command options

Where the syntax is as following:

Keyword	Meaning
src	The source address to be included in the rule. An IP address in dotted-decimal-format, e.g. 64.231.1.10.
src-wildcard	A wildcard for the source address. Expressed in dotted-decimal format this value specifies which bits are significant for matching. One-bits in the wildcard indicate that the corresponding bits are ignored. An example for a valid wildcard is 0.0.0.255, which specifies a class C network.
any	Indicates that IP traffic to or from all IP addresses is to be included in the rule.
host *src*	The address of a single source host.
dest	The destination address to be included in the rule. An IP address in dotted-decimal-format, e.g. 64.231.1.10
dest-wildcard	A wildcard for the destination address. See *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 14, "Link scheduler configuration" on page 148.
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).

```
IPLink(cfg)#profile acl WanRx
IPLink(pf-acl)[WanRx]#deny icmp any any type 8 code 0
IPLink(pf-acl)[WanRx]#exit
IPLink(cfg)#
```

The same effect can also be obtained by using the simpler message name option. See the following example.

```
IPLink(cfg)#profile acl WanRx
IPLink(pf-acl)[WanRX]#deny icmp any any msg echo
IPLink(pf-acl)[WanRX]#exit
IPLink(cfg)#
```

Adding a TCP, UDP or SCTP filter rule to the current access control list profile

The commands **permit** or **deny** are used to define a TCP, UDP or SCTP filter rule. Each TCP, UDP or SCTP filter rule represents a respective access of control list entry.

This procedure describes how to create a TCP, UDP or SCTP access control list entry that permits access

Mode: Profile access control list

Step	Command	Purpose
1	***node*(pf-acl)**[*name*]**#permit {tcp | udp | sctp}** {*src src-wildcard* **| any | host** *src*} **[{eq** *port* **| gt** *port* **| lt** *port* **| range** *from to*}] {*dest dest-wildcard* **| any | host** *dest*} **[{eq** *port* **| gt** *port* **| lt** *port* **| range** *from to*}] **[{cos** *group* **| cos-rtp** *group-data group-ctrl*}]	Creates a TCP, UDP or SCTP access of control list entry that permits access defined according to the command options

This procedure describes how to create a TCP, UDP or SCTP access control list entry that denies access

Mode: Profile access control list

Step	Command	Purpose
1	***node*(pf-acl)**[*name*]**#deny {tcp | udp | sctp}** {*src src-wildcard* **| any | host** *src*} **[{eq** *port* **| gt** *port* **| lt** *port* **| range** *from to*}] {*dest dest-wildcard* **| any | host** *dest*} **[{eq** *port* **| gt** *port* **| lt** *port* **| range** *from to*}] **[{cos** *group* **| cos-rtp** *group-data group-ctrl*}]	Creates a TCP, UDP or SCTP access of control list entry that denies access defined according to the command options

Where the syntax is:

Keyword	Meaning
src	The source address to be included in the rule. An IP address in dotted-decimal-format, e.g. 64.231.1.10.
src-wildcard	A wildcard for the source address. Expressed in dotted-decimal format this value specifies which bits are significant for matching. One-bits in the wildcard indicate that the corresponding bits are ignored. An example for a valid wildcard is 0.0.0.255, which specifies a class C network.
any	Indicates that IP traffic to or from all IP addresses is to be included in the rule.
host src	The address of a single source host.
eq port	Optional. Indicates that a packets port must be equal to the specified port in order to match the rule.
lt port	Optional. Indicates that a packets port must be less than the specified port in order to match the rule.
gt port	Optional. Indicates that a packets port must be greater than the specified port in order to match the rule
range from to	Optional. Indicates that a packets port must be equal or greater than the specified from port and less than the specified to port to match the rule.
dest	The destination address to be included in the rule. An IP address in dotted-decimal-format, e.g. 64.231.1.10.
dest-wildcard	A wildcard for the destination address. See *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 14, "Link scheduler configuration" on page 148.
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.

```
IPLink(cfg)#profile acl WanRx
IPLink(pf-acl)[WanRx]#permit tcp any host 193.14.2.10 eq 80
IPLink(pf-acl)[WanRx]#permit udp host 62.1.2.3 host 193.14.2.11 range 1024 2048
IPLink(pf-acl)[WanRx]#exit
IPLink(cfg)#
```

Binding and unbinding an access control list profile to an IP interface

The command **use** is used to bind an access control list profile to an IP interface. This procedure describes how to bind an access control list profile to incoming packets on an IP interface

Mode: Profile access control list

Step	Command	Purpose
1	***node***(**if-ip**)[*if-name*]**#use profile acl** *name* **in**	Binds access control list profile name to incoming packets on IP interface *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.

```
IPLink(cfg)#context ip router
IPLink(cfg-ip)[router]#interface wan
IPLink(cfg-if)[wan]#use profile acl WanRx in
```

Unbind an access control list profile from an interface.

```
IPLink(cfg)#context ip router
IPLink(cfg-ip)[router]#interface wan
IPLink(cfg-if)[wan]#no use profile acl in
```

Note When unbinding an access control list profile the *name* argument is not required, since only one incoming and outgoing access control list can be active at the same time on a certain IP interface.

Displaying an access control list profile

The **show profile acl** command displays the indicated access control list profile. If no specific profile is selected all installed access control list profiles are shown. If an access control list is linked to an IP interface the number of matches for each rule is displayed. If the access control list profile is linked to more than one IP interface, it will be shown for each interface.

This procedure describes how to display a certain access control list profile

Mode: Administrator execution or any other mode, except the operator execution mode

Step	Command	Purpose
1	***node*#show profile acl** *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.

```
IPLink#show profile acl WanRx
IP access-list WanRx. Linked to router/wan/in.
    deny icmp any any msg echo
    permit ip 62.1.2.3 0.0.255.255 host 193.14.2.11
    permit ip 97.123.111.0 0.0.0.255 host 193.14.2.11
    permit tcp any host 193.14.2.10 eq 80
    permit udp host 62.1.2.3 host 193.14.2.11 range 1024 2048
    deny ip any any
```

Debugging an access control list profile

The **debug acl** command is used to debug the access control list profiles during system operation. Use the **no** form of this command to disable any debug output.

This procedure describes how to debug the access control list profiles

Mode: Administrator execution or any other mode, except the operator execution

Step	Command	Purpose
1	**node#debug acl**	Enables access control list debug monitor

This procedure describes how to activate the debug level of an access control list profiles for a specific interface.

Mode: Interface

Step	Command	Purpose
1	***node*(cfg)#context ip router**	Selects the IP router context
2	***node*(ctx-ip)[router]#interface** *if-name*	Selects IP interface *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.

```
IPLink(cfg)#context ip router
IPLink(cfg-ip)[router]#interface wan
IPLink(cfg-if)[wan]#debug acl in 7
```

The following example enables the debug monitor for access control lists globally.

```
IPLink#debug acl
```

The following example disables the debug monitor for access control lists globally.

```
IPLink#no debug acl
```

Examples

Denying a specific subnet

Figure 34 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 IPLink device. To prevent access, an incoming filter rule named *Jamming* is defined, which blocks any IP traffic from network 172.16.2.0 and has to be bound to IP interface *lan*.

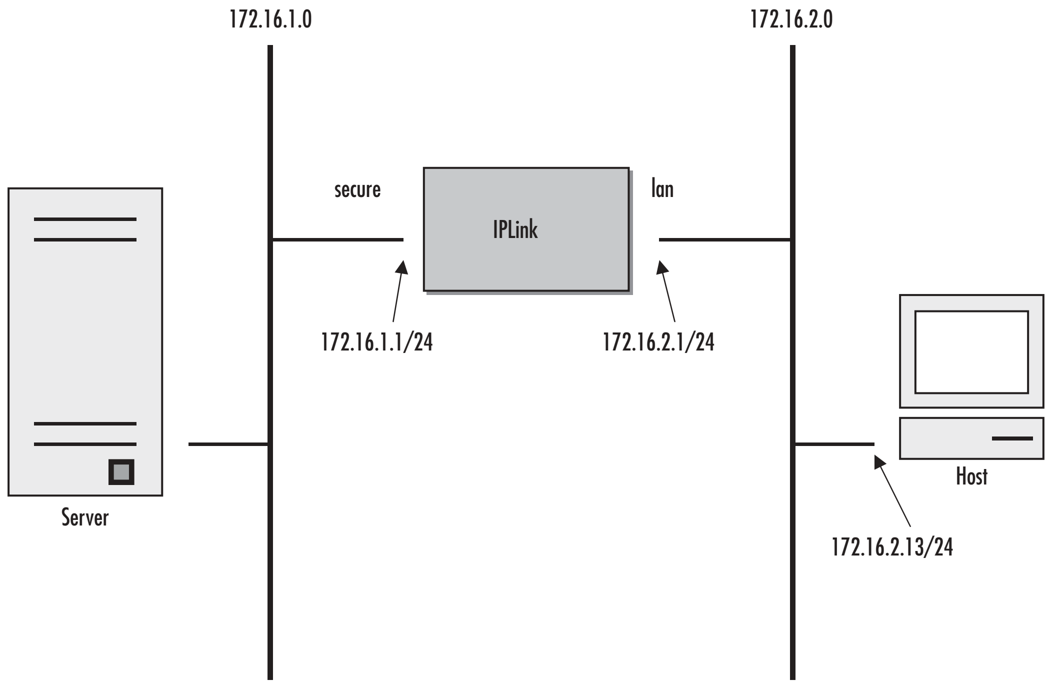


Figure 34. Deny a specific subnet on an interface

The commands that have to be entered are listed below. The commands access the IPLink device via a Telnet session running on a host with IP address 172.16.2.13, which accesses the IPLink 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 20 **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 IPLink software.

This chapter includes the following sections:

- Simple Network Management Protocol (SNMP)
- SNMP tools (see page 228)
- SNMP configuration task list (see page 228)
- Using the AdventNet SNMP utilities (see page 234)
- Standard SNMP version 1 traps (see page 237)

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 IPLink software 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 IPLink devices via SNMP

All IPLink devices have assigned sysObjectID (.iso.org.dod.internet.mgmt.mib-2.system.sysObjectID) numbers (see table 11).

Table 11. IPLink Models and their Unique sysObjectID

IPLink Model	SysObjectID
2802	.iso.org.dod.internet.private.enterprises.patton.products.sn2802 1.3.6.1.4.1.1768.2.2.8.1
2805	.iso.org.dod.internet.private.enterprises.patton.products.sn2805 1.3.6.1.4.1.1768.2.2.8.2
2821	.iso.org.dod.internet.private.enterprises.patton.products.sn2821 1.3.6.1.4.1.1768.2.2.8.3
2835	.iso.org.dod.internet.private.enterprises.patton.products.sn2835 1.3.6.1.4.1.1768.2.2.8.4

According to table 11, an SNMP get request to *.iso.org.dod.internet.mgmt.mib-2.system.sysObjectID* of an IPLink 2802 device reads out a numeric OID of *1.3.6.1.4.1. 1768.2.2.8.1*, which represents an IPLink 2802. The mapping of the sysObjectID to each of the IPLink model is realized with the IPLink product identification MIB.

IMPORTANT

The SNMP agent running in IPLink software is SNMP version 1 (SNMPv1) compliant. SNMP version 2 (SNMPv2) and SNMP version 3 (SNMPv3) are not currently supported.

SNMP tools

Patton recommends the AdventNet MibBrowser, TrapViewer and other SNMP tools. Check the AdventNet Web server at http://www.adventnet.com for latest releases.

Refer to section "Using the AdventNet SNMP utilities" on page 234 for more detailed information on how to use these tools together with the IPLink's.

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 229)
- Setting access community information (required) (see page 231)
- Setting allowed host information (required) (see page 232)
- Specifying the default SNMP trap target (optional) (see page 232)
- Displaying SNMP related information (optional) (see page 233)

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 IPLink devices 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 IPLink, together with information on how to contact that person.

Assigning explanatory location information to describe the system physical location of your IPLink (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 IPLink is accessed via Telnet, is replaced with the hostname.

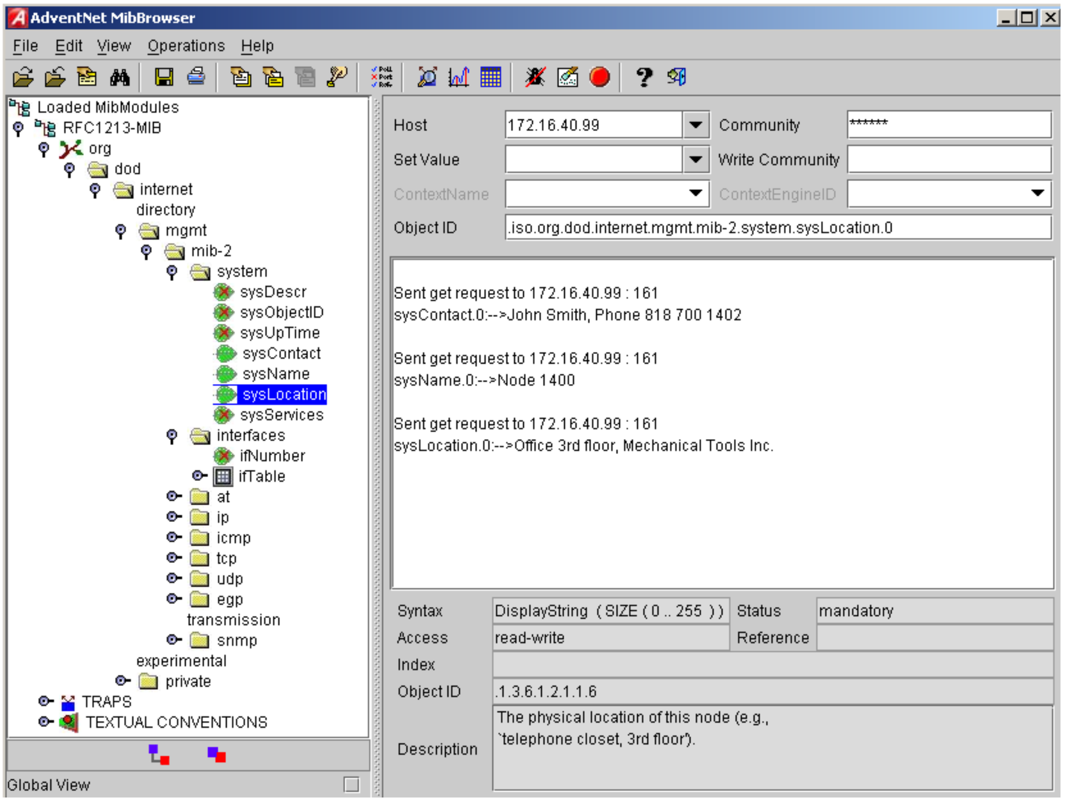
The MIB-II system group values are accessible for reading and writing via the following SNMP objects:

- .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 35.

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 35. 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 35 for a typical MIB browser application accessing these MIB-II system group objects representing the system information.

```
IPLink>enable
IPLink#configure
IPLink(cfg)#system contact "Lorenz Born, Phone 533"
IPLink(cfg)#system location "Office, 3rd floor, Patton Electronics Co."
IPLink(cfg)#system hostname "2803-01"
2803-01(cfg)#
```

After entering a host name the prompt on the CLI no longer displays the IP address of the Ethernet port over which the Telnet session is running but shows the newly entered host name.

Setting access community information

SNMP uses one or more labels called *community strings* to delimit groups of *objects* (variables) that can be viewed or modified on a device. The SNMP data in such a group is organized in a tree structure called a Management Information Base (MIB). A single device may have multiple MIBs connected together into one large structure, and various community strings may provide read-only or read-write access to different, possibly overlapping portions of the larger data structure. An example of a read-only variable might be a counter showing the total number of octets sent or received through an interface. An example of a read-write variable might be the speed of an interface, or the hostname of a device.

Community strings also provide a weak form of access control in earlier versions of SNMP version 1 and 2. SNMP version 3 provides much improved access control using strong authentication and should be preferred over SNMP version 1 and 2 wherever it is supported. If a community string is defined, then it must be provided in any basic SNMP query if the requested operation is to be permitted by the device. Community strings usually allow read-only or read-write access to the entire device. In some cases, a given community string will be limited to one group of read-only or read-write objects described in an individual MIB.

In the absence of additional configuration options to constrain access, knowledge of the single community string for the device is all that is required to gain access to all objects, both read-only and read-write, and to modify any read-write objects.

Note Security problems can be caused by unauthorized individuals possessing knowledge of read-only community strings so they gain read access to confidential information stored on an affected device. Worse can happen if they gain access to read-write community strings that allow unauthorized remote configuration of affected devices, possibly without the system administrators being aware that changes are being made, resulting in a failure of integrity and a possible failure of device availability. To prevent these situations, define community strings that only allow read-only access to the MIB objects should be the default.

By default SNMP uses the default communities *public* and *private*. You probably do not want to use those, as they are the first things an intruder will look for. Choosing community names is like choosing a password. Do not use easily guessed ones; do not use commonly known words, mix letters and other characters, and so on. If you do not intend to allow anyone to use SNMP write commands on your system, then you probably only need one community name.

This procedure describes how to define your own SNMP community

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#snmp community** *name* **{ ro | 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 IPLink device.

```
2803-01(cfg)#snmp community public ro
2803-01(cfg)#snmp community Not4evEryOne rw
```

Note If no community is set on your IPLink 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 IPLink device. Therefore a host needs an entry on an IPLink 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 IPLink 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 IPLink device using community *public* as security name.

```
2803-01(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 an IPLink 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 an IPLink 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-of-hostanme *node* that receives trap messages of this IPLink 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.

```
2803-01(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 an IPLink 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.

```
2803-01(cfg)#show snmp

SNMP Information:
  hostname : 2803-01
  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 an IPLink device. The following tools are the most useful part of the product for IPLink 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 36 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 IPLink device management.

The right frame has labeled text fields to specify the basic parameters like host, community etc. and a Result text area display to view the results.

There are three ways in which the primary window of the MibBrowser can be viewed. It can be viewed with the result display, MIB description panel or multi-variable bind panel in the right frame. The view can be altered in three ways.

- The desired view can be set by the options provided in the display menu item under the view menu. (View → Display →).
- The other way of altering the view is through the general settings panel in the settings menu item in the edit menu. (Edit →‡ Settings)

- The same can be done through clicking the MibBrowser settings button on the toolbar. See figure 36.

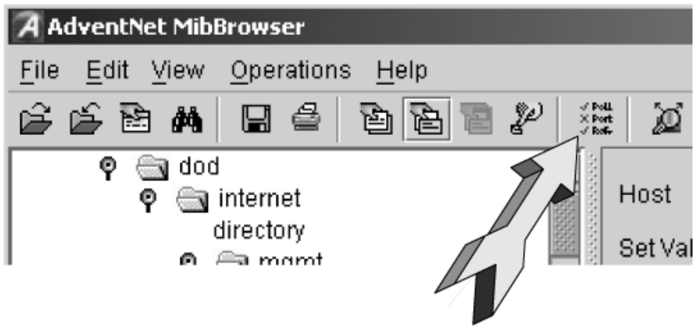


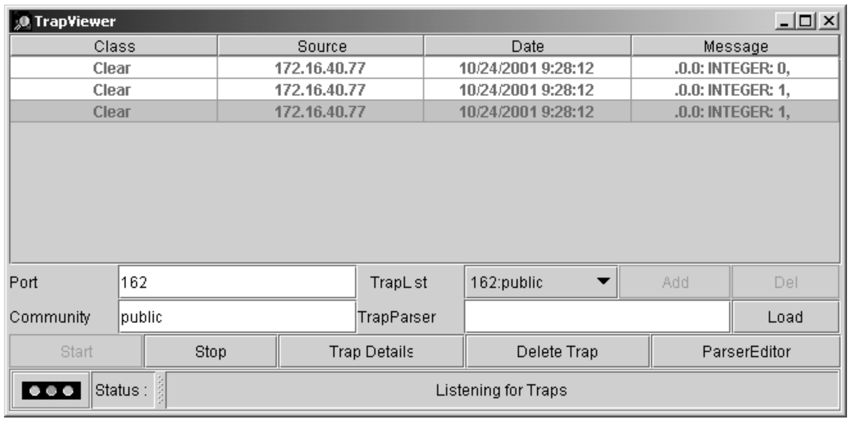
Figure 36. AdventNet MibBrowser Settings Button on the Toolbar

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 an IPLink device. The TrapViewer can listen to one or more port at a time and the traps can be sent from any host. IPLink 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 234. Figure 37 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
Community: public
TrapList: 162:public
TrapParser:
Status : Listening for Traps

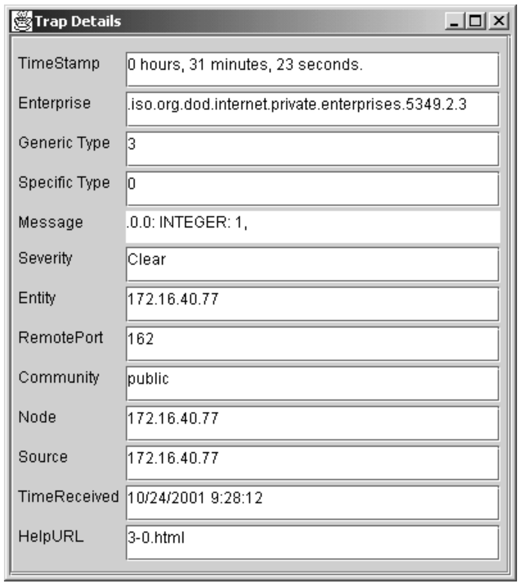
Figure 37. AdventNet TrapViewer displaying received traps

The TrapViewer has a table that displays the trap information, the common parameters text fields where necessary information has to be entered and other options such as Start, Stop, Trap Details, Delete Trap and ParserEditor.

Follow these steps to work on the Trap Viewer and to know more about the available options:

- By default the value in the *Port* text field is 162. Enter the desired port in the field on which the viewer will listen.

- The default value in the *Community* text field is public. Set the community of the incoming traps as desired, depending on the SNMP configuration of your IPLink device.
- Click on *Add* button to add the port and community list on which the trap has to listen to. This is visible in the *TrapList* combo box.
- The port and community list can be deleted by clicking on the *Del* button.
- When you need to load a trap parser file, click on the *Load* button, which will open up a dialog box, from which you can load the parser file.
- In order to receive the traps now, click on the *Start* button. Upon clicking this button, TrapViewer begins to receive traps according to the as-specified port and community.
- Once received, the traps are listed in the trap table of the TrapViewer. By default, the trap table has the following four columns:
  - *Class* that defines the severity of the trap.
  - *Source* that displays the IP address of the source from where the traps were sent.
  - *Date* that shows the date and time when the trap was received.
  - *Message* that by default has the object identifier format (sequence of numeric or textual labels on the nodes along a path from the root to the object) of the trap if any, or it is blank.
- The details of the traps can be viewed by clicking the *Trap Details* button or right click the trap in the trap table and select the option *View Trap Details*. Figure 38 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 38. AdventNet Trap Details window of TrapViewer

The various details available in the Trap Details window are listed in table 12:

Table 12. Details available in the Trap Details window

Trap Details	Description
TimeStamp	The TimeStamp is a 32-bit unsigned value indicating the number of hundredths-of-a-second that have elapsed since the (re)start of the SNMP agent and the sending of the trap. This field shows the value stored in the MIB-II sysUpTime variable converted into hours, minutes and seconds.
Enterprise	This field shows the OID of the management enterprise that defines the trap message. The value is represented as an OBJECT IDENTIFIER value and has a variable length.
Generic Type	The Generic type value is categorized and numbered 0 to 6. They are 0-coldStart, 1-warm-Start, 2-linkDown, 3-linkUp, 4-authenticationFailure, 5-egpNeighborLoss. The trap type value 6 is identified as enterprise-specific value. This field shows the value based on the type of trap.
Specific Type	The specific trap type indicates the specific trap as defined in an enterprise-specific MIB. If the Generic type value is 6 then, this field shows a value greater than 0. If the generic type value is a value other than 6, then the field shows a value 0. This field can have values from 0 to 2147483647.
Message	This is a text field. By default, this field will always contain the Varbinds in the Trap PDU. This can be substituted with text.
Severity	This field shows the Severity or the intensity of the trap. They could be 0-All, 1-Critical, 2-Major, 3-Minor, 4-warning, 5-Clear and 6-info.
Entity	The source IP address from which the Trap was sent is displayed here.
RemotePort	This field reveals the port on which the Trap was sent by the originator.
Community	The Community string is displayed here.
Node	Source
TimeReceived	This displays the Date and Time when the trap was received.
HelpURL	The URL shown here gives more details of the received trap. By default, the URL file name is <generic-type value> - <specific-type value>.html

You can **stop** the listening by clicking the *Stop* button.

When you need to **delete** the trap, select the trap to be deleted and click the *Delete Trap* button or right click on the trap in the trap table and select option *Delete the Selected Rows*.

Yet another option in the Trap Viewer is the *ParserEditor*. The TrapViewer can filter incoming traps according to certain criteria called the parser criteria. The configuration of the criteria is made possible by using the parser **editor**. Refer to the AdventNet SNMP Utilities documentation for a detailed description of the parser editor configuration and its use.

Standard SNMP version 1 traps

IPLink software application software supports the following standard SNMP version 1 traps. The descriptions are taken from RFC 1215 “Convention for defining traps for use with the SNMP”.

```
warmStart TRAP-TYPE
ENTERPRISE snmp
DESCRIPTION
```

```
"A warmStart trap signifies that the sending protocol entity is reinitializing
itself such that neither the agent configuration nor the protocol entity implementa-
tion is altered."
::= 1

linkDown TRAP-TYPE
ENTERPRISE snmp
VARIABLES   { ifIndex }
DESCRIPTION
"A linkDown trap signifies that the sending protocol entity recognizes a failure in
one of the communication links represented in the agent's configuration."
::= 2
```

Note The linkDown trap is not sent if any of the ISDN ports has gone down.

```
linkUp TRAP-TYPE
ENTERPRISE snmp
VARIABLES   { ifIndex }
DESCRIPTION
"A linkUp trap signifies that the sending protocol entity recognizes that one of the
communication links represented in the agent's configuration has come up."
::= 3
```

Note The linkUp trap is not sent if any of the ISDN ports has come up.

```
authenticationFailure TRAP-TYPE
ENTERPRISE snmp
DESCRIPTION
"An authenticationFailure trap signifies that the sending protocol entity is the
addressee of a protocol message that is not properly authenticated. While implemen-
tations of the SNMP must be capable of generating this trap, they must also be capa-
ble of suppressing the emission of such traps via an implementation-specific
mechanism."
::= 4
```

Note The authenticationFailure trap is sent after trying to access any MIB object with a SNMP community string, which does not correspond to the system setting.

```
coldStart TRAP-TYPE
ENTERPRISE snmp
DESCRIPTION
"A coldStart trap signifies that the sending protocol entity is reinitializing
itself such that the agent's configuration or the protocol entity implementation may
be altered."
::= 0
```

Note The standard SNMP version 1 trap coldStart as listed below is *not* supported. After powering up an IPLink device sends a warmStart trap message if any trap target host is defined.

SNMP interface traps

The IPLink sends Interface Traps (*linkUp*, *linkDown*) when the status of logical or physical interfaces change. Logical interfaces are interfaces defined in the IP context (IP interfaces). Physical interfaces are ports in the IPLink terminology (Ethernet, ISDN, and Serial Ports).

The IPLink 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.

```
IPLink(cfg)#show snmp-if-alias-mapping

ifIndex :         Interface Name    (Interface Type) Interface Status
---------------------------------------------------------------------
      1 : ETH00   (ethernet-csmacd)         up
      2 : ETH01   (ethernet-csmacd)         up
      3 : eth00               (IP)          up
      4 : eth01               (IP)          up
```

Interface names in capital letters denote physical interfaces and in small letters logical interfaces.

The IPLink adds an entry to event log for each Interface Traps it sends:

```
IPLink(cfg)#show log

...
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.
...
```

Chapter 21 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 248)

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 242)
- Defining SNTP client operating mode (see page 242)
- Defining SNTP local UDP port (see page 243)
- Enabling and disabling the SNTP client (see page 244)
- Defining the SNTP client anycast address (see page 245)
- Defining SNTP client constant offset to GMT (see page 244
- Enabling and disabling local clock offset compensation (see page 246)
- Defining SNTP client poll interval (see page 244)
- Showing SNTP client related information (see page 247)
- Debugging SNTP client operation (see page 247)

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** *host*	Enter the SNTP primary server IP address or hostname
2	***node***(cfg)**#sntp-client server secondary** *host*	Enter the SNTP secondary server IP address or hostname

Example: Selecting SNTP time servers

In the following example an internal SNTP time server (172.16.1.10) is selected as primary and utcnist.colorado.edu (128.138.140.44) as secondary SNTP time server.

```
IPLink(cfg)#sntp-client server primary 172.16.1.10
IPLink(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 section "Defining the SNTP client anycast address" on page 245 for more details.

Example: Configuring SNTP client operating mode

Configures the SNTP client operating mode to unicast operation

```
IPLink(cfg)#sntp-client operating-mode unicast
```

Configures the SNTP client operating mode to anycast operation

```
IPLink(cfg)#sntp-client operating-mode anycast
```

Configures the SNTP client operating mode to multicast operation

```
IPLink(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 IPLink) 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 (IPLink) 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

```
IPLink(cfg)#sntp-client local-port 123
```

Enabling and disabling the SNTP client

The SNTP client is disabled by default and has to be enabled if clock synchronization shall be used. This procedure describes how to enable or disable the SNTP client

Mode: Configure

Step	Command	Purpose
1	***node***(cfg)#[no] sntp-client	Enables the SNTP client operation. Using the no command syntax disables this feature.

Example: Enabling the SNTP client operation

```
IPLink(cfg)#sntp-client
```

Example: Disabling the SNTP client operation

```
IPLink(cfg)#no sntp-client
```

Defining SNTP client poll interval

Specifies the seconds between each SNTP client request in unicast or anycast mode.

This SNTP client poll interval can be defined to be within the range from 1 to 4’294’967’295. The default value is 60 seconds.

This procedure describes how to set the SNTP client poll interval

Mode: Configure

Step	Command	Purpose
1	***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.

```
IPLink(cfg)#sntp-client poll-interval 30
```

Defining SNTP client constant offset to GMT

Setting the offset of the IPLink 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.

```
IPLink>enable
IPLink#configure
IPLink(cfg)#sntp-client gmt-offset + 02:00:00
```

There is a short form notation supported as shown in the following example.

```
IPLink(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 IPLink. An anycast client (IPLink) 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 IPLink sends a request to a designated local broadcast or multicast group address and expects a reply from one or more anycast servers. The IPLink 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.

```
IPLink(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

```
IPLink(cfg)#sntp-client root-delay-compensation
```

Example: Disabling the SNTP client root delay compensation

```
IPLink(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

```
IPLink(cfg)#show sntp-client
-------------------------------------------
SNTP client        enabled
Operating mode     unicast
Local port         123
Primary server     172.16.1.10:123 v4
Secondary server   128.138.140.44:123 v4
Anycast address    224.0.1.1:123
Poll interval      30sec
Local clock offset disabled
GMT offset         +2:00:00
-------------------------------------------
```

Debugging SNTP client operation

During setup and operation, debugging the behavior of the SNTP client is very useful.

Note The debug sntp client is only available in superuser mode.

This procedure describes how to enable or disable debugging

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#debug sntp client**	Enables and disables SNTP debug monitor. Using the no command syntax disables this feature.

Example: Enable the SNTP debug monitor

The following example shows how to enable the SNTP debug monitor and some typical debug information.

```
IPLink(cfg)#debug sntp client
IPLink(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.

```
IPLink(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 13 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 13. 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
time-c.timefreq.bldrdoc.gov	132.163.4.103	1	Colorado

Table 13. Time servers operated by NIST (Continued)

Server Name	IP Address	Note	Location
time-d.timefreq.bldrdoc.gov	132.163.4.104	3	Colorado
time.nist.gov	192.43.244.18	1	Colorado
time-nw.nist.gov	131.107.1.10	1	Washington

Legend

1. Heavily loaded and not recommended for new users.
2. Recommended for new users.
3. Used for testing only. Not for general users.
4. Does not support anonymous ftp connections.

For more information about NIST Internet Time Service (ITS) check their web server at **http://www.boulder.nist.gov/timefreq/service/its.htm**

Other public NTP primary (stratum 1) time servers

Switzerland

- swisstime.ethz.ch (129.132.2.21)
- Location: Integrated Systems Laboratory, Swiss Fed. Inst. of Technology, CH 8092 Zurich, Switzerland
- Geographic Coordinates: 47:23N, 8:32E
- Synchronization: NTP primary (DCF77 clock), Sun-4/SunOS 4.1.4
- Service Area: Switzerland/Europe
- Access Policy: open access
- Contact: Christoph Wicki (time@iis.ee.ethz.ch)

Germany

- DE ntp0.fau.de (131.188.34.75)
- Location: University Erlangen-Nuernberg, D-91058 Erlangen, FRG
- Geographic Coordinates: 49.573N 11.028E (from Meinberg GPS 166)
- Synchronization: NTP V3 primary (GPS receiver (<<1us)), Sun SS12/Unix SunOS 5.6
- Service Area: Germany/Europe
- Access Policy: open access, pick one of ntp{0,1,2}.fau.de
- Contact: The Timekeepers (time@informatik.uni-erlangen.de)

Note IP addresses are subject to change; please use DNS

- DE ntp1.fau.de (131.188.34.45)
- Location: University Erlangen-Nuernberg, D-91058 Erlangen, FRG
- Geographic Coordinates: 49.573N 11.028E (from Meinberg GPS 166)

- Synchronization: NTP V3 primary (DCF77 PZF receiver (<50us)), Sun E3000 SunOS 5.6
- Service Area: Germany/Europe
- Access Policy: open access, pick one of ntp{0,1,2}.fau.de
- Contact: The Timekeepers (time@informatik.uni-erlangen.de)

Note IP addresses are subject to change; please use DNS

- DE ntps1-0.cs.tu-berlin.de (130.149.17.21)
- Location: Technische Universitaet Berlin, D-10587 Berlin, FRG
- Geographic Coordinates: 52.518N 13.326E
- Synchronization: NTP V3 primary (Meinberg GPS 166), Sun 4/65 SunOS4.1.3
- Service Area: Germany/Europe
- Access Policy: open access
- Contact: Gerard Gschwind (gg@cs.tu-berlin.de)

Additional information on NTP and a list of other NTP servers

The University of Delaware hosts a World Wide Web site that provides additional information on NTP and a list of other NTP servers that are publicly available around the world. In many cases, Internet service providers, universities, and other institutions also provide NTP servers for their own communities. NTP servers other than the NIST NTP servers (listed above) may or may not be of comparable accuracy, and may or may not satisfy traceability requirements. For more information, please see **http://www.eecis.udel.edu/~ntp/**.

Recommended RFC

RFC2030 “Simple Network Time Protocol (SNTP) Version 4”, is available from the Web server at **http://www.faqs.org/rfcs/rfc2030.html**.

Chapter 22 DHCP configuration

Chapter contents

Introduction

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

- DHCP-client configuration tasks (see page 253)
- DHCP-server configuration tasks (see page 256)

The Dynamic Host Configuration Protocol (DHCP) automates the process of configuring new and existing devices on TCP/IP networks. DHCP performs many of the same functions a network administrator carries out when connecting a computer to a network. Replacing manual configuration by a program adds flexibility, mobility, and control to networked computer configurations.

The tedious and time-consuming method of assigning IP addresses was replaced by automatic distributing IP addresses. The days when a network administrator had to manually configure each new network device before it could be used on the network are past.

In addition to distributing IP addresses, DHCP enables configuration information to be distributed in the form of DHCP options. These options include, for example, the default router address, domain name server addresses, the name of a boot file to load etc.

A new expression in DHCP is lease. Rather than simply assigning each DHCP-client an IP address to keep until the client is done with it, the DHCP-server assigns the client an IP address with a lease; the client is allowed to use the IP address only for the duration of that lease. When the lease expires, the client is forced to stop using that IP address. To prevent a lease from expiring, which essentially shuts down all network access for the client, the client must renew its lease on its IP address from time to time.

In IPLink software, 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 IPLink. 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 39.

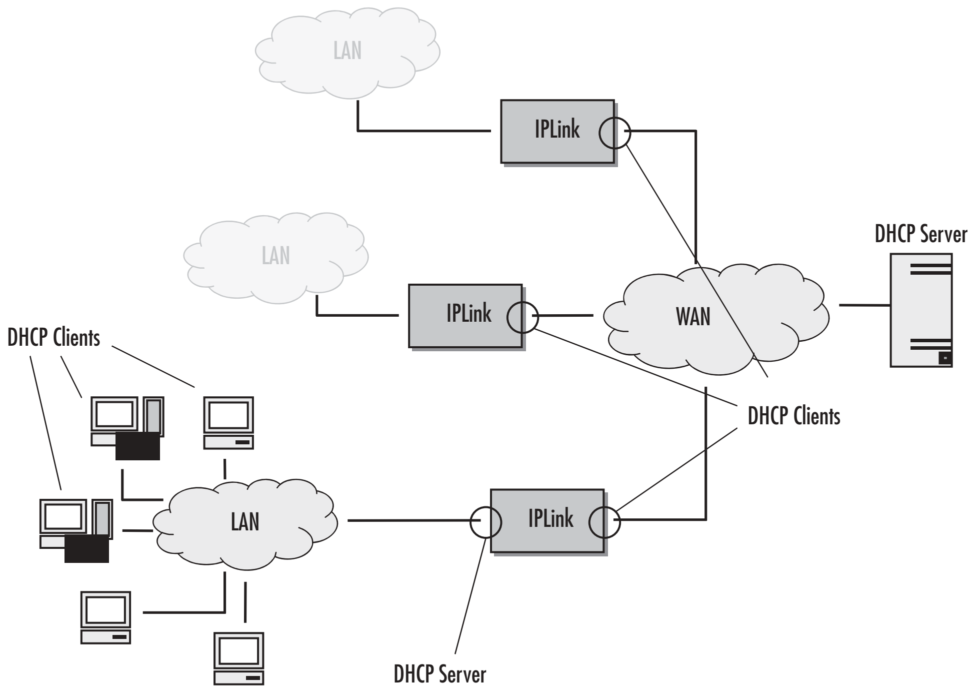


Figure 39. DHCP-client and DHCP-server on the IPLink

DHCP-client configuration tasks

To configure the IPLink 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 255)
- Get debug output from DHCP-client (see page 255)
- Configure DHCP agent

Enable DHCP-client on an IP interface

On every created IP interface a DHCP-client could be enabled. If enabled, the IPLink 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 For example, default gateway, lease expire time, etc.

Note If you are connected to the IPLink by Telnet over the IP interface on which you enable the DHCP-client, the connection is lost after entering the command **ipaddress dhcp**. You need to know the new IP address distributed from the DHCP-server to connect to the IPLink again!

Example: Enable DHCP-client on an IP interface

```
IPLink(cfg)#context ip
IPLink(ctx-ip)[router]#interface eth0
IPLink(if-ip)[eth0]#ipaddress dhcp
IPLink(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 IPLink by Telnet over the IP interface on which you release the DHCP lease, the connection is lost after entering the command **dhcp-client release**. You need an other way (e.g. a serial connection) to connect to the IPLink 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**.

```
IPLink(cfg)#context ip
IPLink(ctx-ip)[router]#interface eth0
IPLink(if-ip)[eth0]#debug dhcp-client
IPLink(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.

IPLink(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 IPLink 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 258)
- Check DHCP-server configuration and status (see page 259)
- Get debug output from the DHCP-server (see page 259)

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: uses broadcasts, h: hybrid - queries the name server first, then broadcasts, m: broadcasts first, the queries the name server, p: only point-to-point name queries to a name server) DHCP Option 46

Step	Command	Purpose
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')
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.

```
IPLink(cfg)#profile dhcp-server LAN
IPLink(pf-dhcps)[lan]#network 192.168.1.0 255.255.255.0
IPLink(pf-dhcps)[lan]#include 192.168.1.32 192.168.1.63
IPLink(pf-dhcps)[lan]#lease 2 days
IPLink(pf-dhcps)[lan]#default-router 192.168.1.1
IPLink(pf-dhcps)[lan]#domain-name-server 80.254.161.125
IPLink(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.

```
IPLink(ctx-ip)[router]#dhcp-server use LAN
IPLink(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:

```
IPLink(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.

```
IPLink(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 23 DNS configuration

Chapter contents

Introduction

The domain name system (DNS) enables users to contact a remote host by using easily remembered text labels (www.patton.com, for example) instead of having to use the host's numeric address (209.45.110.15, for example). When DNS names are entered as part of configuration commands or CLI exec mode commands in applications like Ping, Traceroute, or Tftp, the IPLink uses a DNS resolver component to convert the DNS names into the numeric address.

The IPLink can be configured as a caching DNS relay server to speed data transfers, acting as the DNS server for a private network. In this configuration, hosts in the network send their DNS queries to the IPLink, which checks to see if the DNS name is in its DNS resolver cache. If it finds the name in cache, the IPLink uses the cached data to resolve the DNS name into a numeric IP address. If the name is not in cache, the query is forwarded on to a DNS server. When the IPLink receives the answer from the server, it adds the name to the cache, and forwards it on to the host that originated the query. This process enables the IPLink to provide answers more quickly to often-queried DNS names, reducing the number of DNS queries that must be sent across the access link.

DNS configuration task list

The following sections describe how to configure the DNS component:

- Enabling the DNS resolver
- Enabling the DNS relay

Enabling the DNS resolver

To enable the IPLink DNS resolver, you must configure it with the address of one or more DNS servers that will be used to resolve DNS name queries. If multiple DNS servers are configured, the IPLink will query each server in turn until a response is received. DNS servers are configured as follows:

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#dns domain-name server** *server-ip-address*	Add an IP address of a DNS server to be used resolving DNS names
2		Repeat step 1 for each additional DNS server you want to add
2	***node*(cfg)#dns-client cache** *number-of-entries*	Optional. Defines the maximum number of DNS answers stored within the cache (default is 30)

Example: Configuring DNS servers

The following example shows how to add DNS servers to the IPLink DNS resolver and increase the size of the DNS cache to 100 entries.

```
IPLink>enable
IPLink#configure
IPLink(cfg)#dns-client server 62.2.32.5
IPLink(cfg)#dns-client server 62.2.100.45
IPLink(cfg)#dns-client cache 100
```

You can test the DNS server configuration using the **dns-lookup** command as follows:

Example: Testing DNS server configuration

```
IPLink(cfg)#dns-lookup www.patton.com
Name:    www.patton.com
Address: 209.49.110.5
```

Note The DNS resolver automatically learns domain name servers if it receives them through PPP or DHCP protocols. You can verify that the DNS resolver has received domain name servers by using the **show dns-client** command as follows:

```
IPLink(cfg)#show dns-client
The following DNS servers are currently available:
  Configured IP: 195.186.1.110
  Discovered IP: 81.221.250.10  (Not used)
  Discovered IP: 81.221.252.10  (Not used)
IPLink(cfg)#
```

Configured IP indicates a domain name server that has been configured as shown at the beginning of this section. *Discovered IP* indicates a domain name server that was learned automatically.

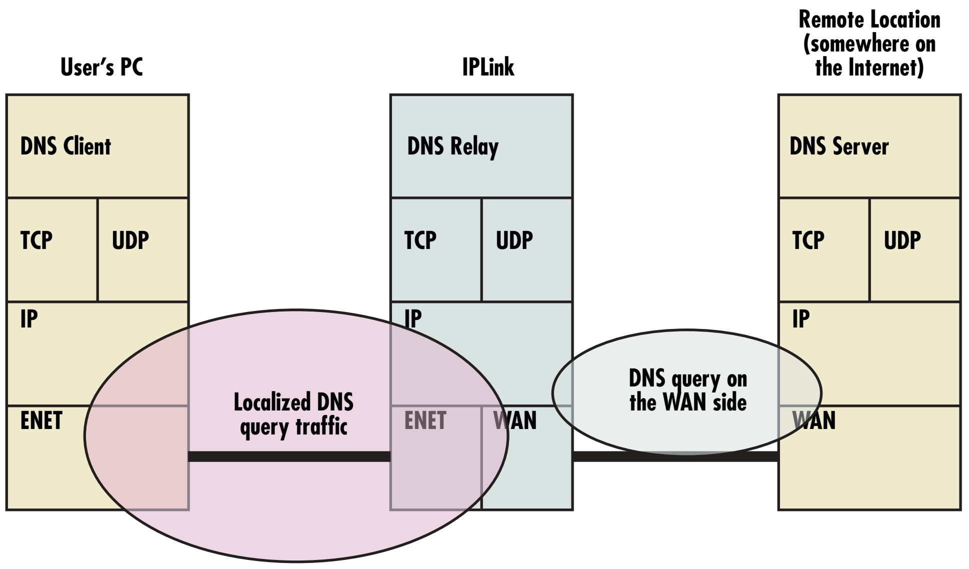

User's PC
IPLink
Remote Location (somewhere on the Internet)
DNS Client
TCP
UDP
IP
ENET
DNS Relay
TCP
UDP
IP
ENET
WAN
DNS Server
TCP
UDP
IP
WAN
Localized DNS query traffic
DNS query on the WAN side

Figure 40. DNS relay diagram

Enabling the DNS relay

DNS (Domain Name System) is a distributed database used in IP networks to provide the numerical IP address for a URL's host name. There are DNS Servers, DNS Relays, and DNS Clients (see figure 40). DNS clients send queries with the host-name of interest to the DNS Server. The DNS server responds with the IP address. DNS Relay agents maintain a cache of host names and IP addresses, much smaller than a DNS Server. It acts as a liaison between the DNS Server and the DNS client

Advantages in configuring a DNS Relay in the IPLink are:

- Network traffic is reduced since only a single query is sent to the DNS server although numerous users may be requesting an IP address for the same host name
- The DNS queries are localized between the Users and the IPLink which reduces congestion on the WAN side of the IPLink
- Multiple DNS servers can be consulted from the IPLink

The DNS resolver must be configured before you can use the DNS relay feature (see section "Enabling the DNS resolver" on page 262 to enable the DNS resolver, if you have not already done so).

Do the following to enable the DNS relay feature:

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#dns-relay**	Enables DNS relay feature

Example: Enabling DNS relay

The following example shows how to enable the DNS relay feature.

```
IPLink>enable
IPLink#configure
IPLink(cfg)#dns-relay
```

Note If a DHCP server profile has been set up, you can announce the IPLink as domain name server to the DHCP clients as follows:

```
IPLink(cfg)#profile dhcp-server LAN
IPLink(pf-dhcps)[LAN]#domain-name-server <ip-address>
```

Where *ip-address* must be the IP address of the IPLink IP interface to which the DHCP clients are connected.

Chapter 24 DynDNS configuration

Chapter contents

Introduction

IPLink devices are often used in applications where the addresses of their IP interfaces are not assigned statically (i.e. permanently) but instead are configured dynamically. In these applications, the IP address is assigned dynamically using protocols like DHCP or PPP. The problem with dynamically assigning addresses is that when the IP address changes, remote devices can no longer contact the IPLink because they do not know what the new address is.

Dynamic DNS (DynDNS) addresses this problem by registering a permanent hostname for your IPLink. DynDNS then directs traffic sent to the registered host name on to the IPLink's ISP-assigned dynamic IP address, enabling the IPLink to be accessed from the Internet without knowing its current dynamic IP address.

The DNS server used for registration is operated by Dynamic Network Services, Inc. You can find detailed information about the company and the services it offers on the webpage www.dyndns.org. The company offers different levels of service. The basic services are offered free of charge, while the more advanced services are chargeable.

The IPLink supports the following DynDNS services:

- Dynamic DNS
- Static DNS
- Custom DNS

DynDNS configuration task list

This section describes configuring the DynDNS service. All possible configurations, which are involved in a specific configuration topic are described in the respective configuration task. To get a minimal working configuration of the DynDNS client, you must execute all the configuration tasks of the list below, except the tasks explicitly marked as optional.

- Creating a DynDNS account
- Configuring the DNS resolver
- Configuring basic DynDNS settings
- Configuring advanced DynDNS settings (optional)

Creating a DynDNS account

Before using the DynDNS service, you must create a DynDNS account on the DynDNS server and add a hostname to your account, which can be updated by the IPLink. Go to the DynDNS website at www.dyndns.org and follow the instructions on the webpage to create the account and add a hostname.

Configuring the DNS resolver

The DynDNS client requires that the IPLink's DNS resolver be enabled. You can find additional information about how to configure the DNS resolver in chapter 23, "DNS configuration" on page 261.

Configuring basic DynDNS settings

The following procedure describes the steps necessary to enable the DynDNS feature.

Mode: DynDNS

Step	Command	Purpose
1	**node(dyndns)#authentication** *user password*	Defines the authentication credentials of your DynDNS account
2	**node(dyndns)#service {dynamic|static|custom}**	Defines the DynDNS service to use
3	**node(dyndns)#hostname** *name*	Defines the hostname that will be assigned to the IPLink
4	**node(dyndns)#observe** *ip-interface-name*	Defines the IP interface to observe for IP address changes. When the IP address on this interface changes, the hostname to IP address mapping on the DynDNS server will be updated

Example: Configuring DynDNS

The following example shows the necessary steps required for a basic working configuration of the DynDNS client.

```
IPLink>enable
IPLink#configure
IPLink(cfg)#context ip
IPLink(ctx-ip)[router]#dyndns
IPLink(dyndns)#authentication Bob 245gf46te
IPLink(dyndns)#service dynamic
IPLink(dyndns)#hostname myhostname.dyndns.org
IPLink(dyndns)#observe eth1
```

Configuring advanced DynDNS settings (optional)

Defining a mail exchanger for your hostname

If required, you can define a mail exchanger or a backup mail exchanger for your hostname on the DynDNS server.

Mode: DynDNS

Step	Command	Purpose
1	**node(dyndns)# mail-exchanger** *hostname* **[backup-mx]**	Defines the host, which is the mail exchanger for your hostname. If the backup-mx parameter is specified, the mail-exchanger will be registered as backup mail exchanger only

Example: Defining a mail exchanger

The following example shows how to define a mail exchanger named *mail.mycompany.com*, which should be used as the primary mail-exchanger for the registered DynDNS hostname.

```
IPLink>enable
IPLink#configure
IPLink(cfg)#context ip
IPLink(ctx-ip)[router]#dyndns
IPLink(dyndns)#mail-exchanger mail.mycompany.com
```

Troubleshooting

The DynDNS component provides several commands to analyze and solve DynDNS problems. You can retrieve basic DynDNS status information as follows:

Mode: DynDNS

Step	Command	Purpose
1	**node(dyndns)#show dyndns**	Display basic DynDNS status information

Example: Displaying DynDNS status information

The following example displays status information of a properly configured and working DynDNS client.

```
IPLink(dyndns)#show dyndns
              Current state: Idle
              Last registered address: 243.232.39.64
              Hostname: test.dyndns.org
```

You can also monitor current activities of the DynDNS client. This includes ongoing DNS queries for DynDNS servers, verification of the currently registered IP address and updating the registration on the DynDNS server. The debug monitor can be enabled as follows;

Mode: Configure

Step	Command	Purpose
1	**node(cfg)#debug dyndns**	Enable the DynDNS debug monitor

Example: Displaying DynDNS status information

The following example shows how to enable the debug monitor and the output of the monitor when the IP address on the DynDNS server can be updated successfully.

```
IPLink(dyndns)#debug dyndns
            16:20:43  DYNDNS> Resolving 'checkip.dyndns.org'...
            16:20:43  DYNDNS> Resolved 'checkip.dyndns.org'.
            16:20:43  DYNDNS> Retrieving current IP address...
            16:20:43  DYNDNS> Sending request...
            16:20:43  DYNDNS> Current IP address (57.32.59.64) does not match last
registered one
                                         (43.23.44.2). DNS update is required.
            16:20:43  DYNDNS> Resolving 'update.dyndns.org'...
```

```
16:20:43  DYNDNS> Resolved 'update.dyndns.org'.
16:20:43  DYNDNS> Updating DNS...
16:20:43  DYNDNS> Sending request...
16:20:44  DYNDNS> DNS updated successfully
16:20:44  DYNDNS> Registered IP address is (57.32.59.64).
```

If required, you can force the DynDNS component to re-register the current IP address on the DynDNS server—even if the dynamic IP address has not changed—using the following command (this command could also be useful for observing the update process in the debug monitor).

IMPORTANT

Possible blocking—Do not use this command too often, because the DynDNS server will block your hostname, if you trigger too many unnecessary updates of your IP address.

You can also force the DynDNS client to resume normal operation, if the state of the DynDNS client is shown as blocked and the problem which led to the blocked state has been solved. The DynDNS client will enter the blocked state if the DynDNS server reports an unrecoverable error during DNS updates that require user intervention. These are mainly configuration problems, such as invalid credentials or an invalid hostname.

Mode: DynDNS

Step	Command	Purpose
1	**node(dyndns)#dyndns reset**	Forces a re-registration of the current IP address on the DynDNS server, even if an update is not necessary

Chapter 25 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 etc. IPLink software offers PPP over the following link layers:

- PPP over Ethernet (PPPoE)
- PPP over serial link, i.e. V.35/X.21, HDLC

Figure 41 shows the elements involved in the configuration of PPP. The elements required to configure PPP over Ethernet are located in the upper left corner of the figure. The elements for PPP over serial link are in the lower left corner.

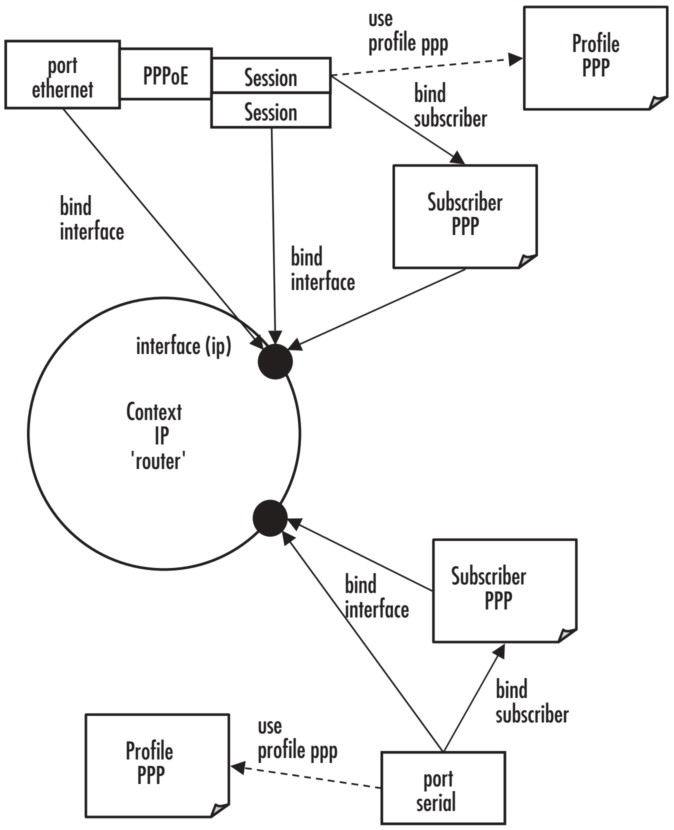


Figure 41. PPP configuration overview

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
- Configuring for IP address auto-configuration from PPP (see page 274)
- Creating a PPP subscriber (for authentication) (see page 274)
- Configuring a PPPoE session (see page 276)
- Configuring a serial port for PPP (see page 278)
- Creating a PPP interface within the CS context (not currently available)
- Creating a PPP profile (see page 279)
- Displaying PPP configuration information (see page 280)
- Debugging PPP (see page 281)

Creating an IP interface for PPP

An IP interface is required to link a PPP connection to the IP context. The IP interface must apply a network address port translation (NAPT) if the PPP service provider only offers a single IP address and not an IP subnet, or if the IP addresses on the LAN shall be private and hidden behind a public IP address (see 12, "NAT/NAPT configuration" on page 128 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 ***node*(if-ip)[name]#ipaddress dhcp** or ***node*(if-ip)[name]# ipaddress** *ip-address netmask*	The PPP remote peer offers an IP address for the IP interface. The IP interface adopts this IP address 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. 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 | 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. PPP over Ethernet connections impose an overhead of 8 Bytes (PPP: 2 Bytes, PPPoE: 6 Bytes). Some Ethernets do not allow payloads larger than the 1500 Bytes which the standard defines, so IP packets must not contain more than 1492 bytes when transmitted over such connections. Reducing the MTU/MRU to 1492 Bytes does not always solve the problem because many sources do not allow fragmentation of the IP packets they send (they set the 'Don't fragment'). However, these sources limit the size of the IP packets according to the MSS which their peers announce in the TCP SYN packets. 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 12, “NAT/NAPT configuration” on page 128 to learn how to create a NAPT profile.

Example: Create an IP interface for PPP

The following procedure creates an IP interface that can be used for all three types of link layers. The command lines **tcp adjust-mss** only apply to Ethernet link layers.

```
IPLink(cfg)#context ip router
IPLink(ctx-ip)[router]#interface ppp_interface
IPLink(if-ip)[ppp_int~]#point-to-point
IPLink(if-ip)[ppp_int~]#ipaddress unnumbered
IPLink(if-ip)[ppp_int~]#tcp adjust-mss rx mtu
IPLink(if-ip)[ppp_int~]#tcp adjust-mss tx mtu
```

Disable interface IP address auto-configuration from PPP

This procedure enables/disables automatic configuration of the interface IP address from the PPP network control protocol negotiation.

Mode: profile ppp

Step	Command	Purpose
1	[*name*] **(pf-ppp)# [no] local-address-autoconfig**	Enables or disables auto-configuration of the local IP address from PPP. Default: enabled.

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. This information allows to use different subscribers for incoming and outgoing calls. Set the direction as follows: • PPP over Ethernet: ‘dial out’ • PPP over Serial: ‘dial in’

Step	Command	Purpose
3	***node*(subscr)[name]# [no] authentication { (chap pap) | {chap|pap} }**	Defines the authentication protocol to be used, PAP and/or CHAP
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 272.

Example: Create a PPP subscriber

The procedure below creates a PPP subscriber for a PAP authentication with some Internet Service Provider.

```
IPLink(cfg)#subscriber ppp joe_example
IPLink(subscr)[joe_exa~]#dial out
IPLink(subscr)[joe_exa~]#authentication pap
IPLink(subscr)[joe_exa~]#identification outbound joeexample@isp.com password
blue4you
IPLink(subscr)[joe_exa~]#bind interface ppp_interface router
```

Trigger forced reconnect of PPP sessions using a timer

In some situations, it is useful to disconnect and reconnect a PPP session at a clearly defined time. The following procedure shows how PPP can be configured to reconnect the connection every time a timer expires.

A common application for this feature: some ISPs disconnect the PPP session after a fixed period of time, for example, 16 hours. This may cause call interruptions if it happens during the day. The timer allows to disconnect and reopen the PPP session at a predefined time, such as 0200 hours.

Mode: subscriber ppp <subscriber>

Step	Command	Purpose
1	[*name*] **(subscr)**[subscriber]**# [no] timeout on-timer** <timer>	Enables/disables forced reconnect every time the timer *<timer>* expiries.

Disable interface IP address auto-configuration from PPP

This procedure enables/disables automatic configuration of the interface IP address from the PPP network control protocol negotiation.

Mode: profile ppp

Step	Command	Purpose
1	[*name*] **(pf-ppp)[no]# [no] local-address-autoconfig**	Enables/disables autoconfiguration of the local IP address from PPP. Default: *enabled*.

Configuring a PPPoE session

PPP can run over Ethernet (PPPoE). The *active discovery* protocol identifies the PPP remote peer on the Ethernet and establishes a PPPoE session with it. The PPPoE session provides a logical point-to-point link that to runs PPP as if it was a physical point-to-point link (e.g. a serial link).

This procedure describes how to configure an Ethernet port and a session for PPPoE

Mode: Configure

Step	Command	Purpose
1	***node***(cfg) **#port ethernet** *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.

```
IPLink(cfg)#port ethernet 0 0
IPLink(prt-eth)[0/0]#encapsulation pppoe
IPLink(prt-eth)[0/0]#no shutdown
IPLink(prt-eth)[0/0]#pppoe
IPLink(pppoe)[0/0]#session green
IPLink(session)[green]#bind subscriber joe_example
IPLink(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 IPLink 2821 or 2835, respectively. 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.

```
IPLink(cfg)#port serial 0 0
IPLink(prt-ser)[0/0]#encapsulation ppp
IPLink(prt-ser)[0/0]#bind subscriber joe_example
IPLink(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.
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.

```
IPLink(cfg)#profile ppp PPPoE
IPLink(pf-ppp)[PPPoE]#mtu min 68 max 1492
IPLink(pf-ppp)[PPPoE]#mru min 68 max 1492
IPLink(pf-ppp)[PPPoE]#van-jacobson compression
IPLink(pf-ppp)[PPPoE]#port ethernet 0 0
IPLink(prt-eth)[0/0]#pppoe
IPLink(pppoe)[0/0]#session green
IPLink(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

```
IPLink(session)[green]#show subscriber ppp joe_example

Subscribers:
------------

Name:                       joe_example
Direction:                  dial-out
Authentication:             pap
Identification (inbound):   (none)
Identification (outbound):  patton/patton
Timeout for disconnect:     no absolute timeout, no idle timeout
Max. sessions:              no limit
IP address:                 (none)
Callback:                   (none)
Binding:                    interface ppp_interface router
Binding:                    interface ppp_interface router
```

Example: Display a PPP profile

```
IPLink(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 IPLink 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

```
IPLink(cfg)#show ppp links 4

PPP Link Information:
=====================
Link:
  ID:                    0
  Name:                  ethernet 0 0 0/pppoe/ppp_green
  Protocols:             LCP, PAP
LCP:
  ID:                    0
  Name:                  ethernet 0 0 0/pppoe/ppp_green
  State:                 Opened
  Conf-Req send rate:    3000ms
  Max. Conf-Req:         10
  Term-Req send rate:    3000ms
  Max. Term-Req:         2
  Echo-Req send rate:    10000ms
  Max. Echo-Req:         3
  Local ID:              100000020390
  Remote ID:
  Local configured options:
    Magic Number = 0x00000000
    MRU = 1492 [68,1492]
    ACCM = 0xffffffff
  Local acknowledged options:
  Remote configured options:
    Magic Number = 0xb89d9e6b
    MRU = 1492 [68,1492]
    ACCM = 0xffffffff
    Authentication Protocol = { PAP }
  Remote acknowledged options:
    MRU = 1492 [68,1492]
    Magic Number = 0xb89d9e6b
    Authentication Protocol = { PAP }
  Remote denied options:
  Remote rejected options:
PAP:
  ID:                    0
  Name:                  ethernet 0 0 0/pppoe/ppp_green
  State:                 Opened
  Direction:             supplying
  Local authentication:
    ID:                  patton
    Password:            patton
    Success:
  Remote authentication:
    ID:
    Password:
    Success:             Greetings!!
  Auth-Req send rate:    3000ms
  Max. Auth-Req:         3
```

Example: Display PPP network protocol information

```
IPLink(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

```
IPLink(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 26 VPN configuration

Chapter contents

Introduction

This chapter describes how to configure the VPN connections between two IPLink devices or between an IPLink and a third-party device.

A *virtual private network* (VPN) is a private data network that uses the public telecommunications infrastructure, maintaining privacy through the use of a tunneling protocol and security procedures.

There are different technologies to implement a VPN. IPLink software applies the *internet protocol security* (IPsec) Architecture (see RFC 2401). The following sections describe the main building blocks of the IPsec architecture as implemented in IPLink software.

Authentication

Authentication verifies the integrity of data stream and ensures that it is not tampered with while in transit. It also provides confirmation about data stream origin.

Two authentication protocols are available:

- Authentication header (AH): protects the IP payload, the IP header, and the authentication header itself
- Encapsulating security payload (ESP): protects the IP payload and the ESP header and trailer, but not the IP header

Two algorithms perform the authentication:

- HMAC-MD5-96: is a combination of the *keyed-hashing for message authentication* (HMAC) and the *message digest version 5* (MD5) hash algorithm. It requires an authenticator of 128-bit length and calculates a hash of 96 bits over the packet to be protected (see RFC 2403).
- HMAC-SHA1-96: is a combination of the (HMAC) and the *secure hash algorithm version 1* (SHA1). It requires an authenticator of 160 bit length and calculates a hash of 96 bits over the packet to be protected (see RFC 2404).

Encryption

Encryption protects the data in transit from unauthorized access. Encapsulating security payload (ESP) is the protocol to transport encrypted IP packets over IP (see RFC 2406).

The following encryption algorithms are available:

	Key Length [Bit]	RFC
DES-CBC (Data Encryption Standard - Cipher Block Chaining)	56	2405
3DES-CBC (Triple Data Encryption Standard - Cipher Block Chaining)	128 or 192[a]	1851
AES-CBC (Advanced Encryption Standard - Cipher Block Chaining)	128, 192, or 256	3268

a. The 3DES algorithm uses only 112 out of the 128 Bit or 168 out of the 192 Bit as key information. Cisco only supports 192 Bit keys with 3DES.

The single DES algorithm no longer offers adequate security because of its short key length (a minimum key length 100 bits is recommended). The AES algorithm is very efficient and allows the fastest encryption. AES with a key length of 128 bits is therefore the recommended algorithm.

Transport and tunnel modes

The mode determines the payload of the ESP packet and hence the application:

- Transport mode: Encapsulates only the payload of the original IP packet, but not its header, so the IPsec peers must be at the endpoints of the communications link.
- A secure connection between two hosts is the application of the transport mode.
- Tunnel mode: Encapsulates the payload and the header of the original IP packet. The IPsec peers can be (edge) routers that are not at the endpoints of the communications link.

  A secure connection of the two (private) LANs, a 'tunnel', is the application of the tunnel mode.

Key management

The current implementation of IPsec in IPLink software works with pre-shared keys (also called *manual keying* or *manual IPsec*) or using Internet Key Exchange (IKE). Keys are manually generated, distributed, and stored as a hexa-decimal string in the startup-configuration of the IPLink and its peer.

Note Depending on the processing hardware applied to *reverse engineering* a DES key, it can take from 3 hours to 3 days to break the key. Thus, for maximum security, DES keys must be manually updated regularly. AES- or 3DES-keys, because they are much more complex, take so much longer to break as to be practically infinite.

The automatically keyed IPSEC connections using the Internet Key Exchange (IKE / RFC2409) protocol that is based on Internet Security Association and Key Management Protocol (ISAKMP / RFC2408) is the other option. IKE supports authentication using pre-shared keys. There is currently no support for authentication using Public Key Infrastructure (PKI) and digital certificates.

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 288
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.

```
IPLink(cfg)#profile ipsec-transform AES_128
IPLink(pf-ipstr)[AES_128]#esp-encryption aes-cbc 128
```

Creating an IPsec policy profile

The IPsec policy profile supplies the keys for the encryption and/or the authenticators for the authentication, the *security parameters indexes* (SPIs), and IP address of the peer of the secured communication. Furthermore, the profile defines which IPsec transformation profile to apply and whether transport or tunnel mode shall be most effective.

The SPI identifies a secured communication channel. The IPsec component needs the SPI to select the suitable key or authenticator. Inbound and outbound channels can have the same SPI, but the channels in the same direction—inbound or outbound—must have unique SPIs. The SPI is not encrypted and can be monitored.

Procedure: To create an IPsec policy profile

Mode: Configure

Step	Command	Purpose
1	***node***(**cfg)#profile ipsec-policy-manual** *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-aauthentication | 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 288 and "Encryption" on page 288) 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.

```
IPLink(cfg)#profile ipsec-policy-manual ToBerne
IPLink(pf-ipsma)[ToBerne]#use profile ipsec-transform AES_128
IPLink(pf-ipsma)[ToBerne]#session-key inbound esp-encryption
1234567890ABCDEF1234567890ABCDEF
IPLink(pf-ipsma)[ToBerne]#session-key outbound esp-encryption
FEDCBA0987654321FEDCBA0987654321
IPLink(pf-ipsma)[ToBerne]#spi inbound esp 1111
IPLink(pf-ipsma)[ToBerne]#spi outbound esp 2222
IPLink(pf-ipsma)[ToBerne]#peer 200.200.200.1
IPLink(pf-ipsma)[ToBerne]#mode tunnel
```

Creating/modifying an outgoing ACL profile for IPsec

An access control list (ACL) profile in the outgoing direction selects which outgoing traffic to encrypt and/or authenticate, and which IPsec policy profile to use. IPsec does not require an incoming ACL.

Note Outgoing and incoming IPsec traffic passes an ACL (if available) twice, once before and once after encryption/authentication. So the respective ACLs must permit the encrypted/authenticated and the plain traffic.

For detailed information on how to set-up ACL rules, see chapter 19, “Access control list configuration” on page 211.

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.

```
IPLink(cfg)#profile acl VPN_Out
IPLink(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
IPLink(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* You can omit this setting if the default route already points to this IP interface or to a next hub reachable via this IP interface, and if there is no other route. Make also sure that the IP router knows how to reach the peer of the secured communication. Usually, a default route does this job.

Example: Activate outgoing ACL and establish route

The following example configures an outgoing ACL profile that interconnects the two private networks 192.168.1/24 and 172.16/16.

```
IPLink(cfg)#context ip router
IPLink(ctx-ip)[router]#interface WAN
IPLink(if-ip)[WAN]#use profile acl VPN_Out out
IPLink(if-ip)[WAN]#context ip router
IPLink(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

```
IPLink(cfg)#show profile ipsec-transform

IPSEC transform profiles:

Name: AES_128
 ESP Encryption: AES-CBC, Key length: 128
```

Example: Display IPsec policy profiles

```
IPLink(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-associ-ations**	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 IPLink.

Example: IPsec Debug Output

```
IPLink(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

```
IPLink(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
```

Key Management (IKE)

As briefly described in the Introduction, key management is done either by pre-shared keys or automatically keyed IPSEC connections usgin the Internet Key Exchange (IKE / RFC 2409). IKE is based on Internet Security Association and Key Management Protocol (ISAKMP / RFC 2408). The IKE module supports authentication using pre-shared keys. There is currently no support for authentication using Public Key Infrastructure (PKI) and digital certificates.

IKE is used to establish a shared secret between two peers, which can be used to derive encryption and/or authentication keys for the exchange of encrypted and or authenticated packets between the peers through an IPSEC connection. IKE also authenticates the two peers to thwart man in the middle attacks. In addition IKE empowers IPSEC to do replay protection to prevent re-injection of previously captured packets into the protected network. Furthermore IKE negotiates a set of cryptographic transforms used by IPSEC for encryption and/or authentication of IP packets. IKE is also responsible for periodic establishment of new session keys for the ISPEC security associations.

To achieve all of this, IKE is split into two phases called MAIN MODE and QUICK MODE.

In MAIN MODE, IKE mutually authenticates the peers, establishes a shared secret between them and negotiates cryptographic transforms in order to create an ISAKMP security association between the two peers. The ISAKMP security association is only used to provide a secure, authenticated and encrypted channel between the peers, which can be used for any further communication.

In QUICK MODE, IKE negotiates all the security parameters like cryptographic transforms, SPIs and sessions keys. They are required for establishing one or more IPSEC security association. All QUICK MODE communication is protected by a previously established ISAKMP security association.

Note The same ISAKMP security association can be used to establish multiple quick modes.

Main differences between manual & IKE IPSEC configurations

- For IKE connections the ACLs must allow traffic from and to UDP port 500 in plaintext, because this port is used by IKE to negotiate security associations.
- The ¨profile ipsec-transform¨ defines the cryptographic transforms used for the IPSEC connections, but it is necessary also to define also a ¨profile isakmp-transform¨, that defines the cryptographic transforms used to protect the negotiation of new IPSEC security associations using ISAKMP.
- Instead of the ¨profile ipsec-policy-manual¨, which is used to create manual keyed IPSEC connections, you need to create a ¨profile ipsec-policy-isakmp¨, which contains all the IKE specific configuration options.

Creating an IPSEC transform profile

First you need to create at least one IPSEC transform profile. In addition to the parameters used also for manually keyed IPSEC security associations, you can optionally also specify a security association lifetime for IKE security associations. If the lifetime of the security association expires, IKE will automatically negotiate a new security association. The default lifetime for ISPEC security associations is one hour without any limit on the transmitted data volume. The parameters defined in this profile are used for the negotiation of IPSEC security associations in quick mode.

The following commands can be used to change the security association lifetime:

Mode: profile ipsec-transform <transform-name>

Step	Command	Purpose
1 optional	***node*(pf-ipstr)[ctx-*name*]# key-lifetime-seconds <seconds>**	Define a new maximum lifetime of the security associations in seconds.
2 optional	***node*(pf-ipstr)[ctx-*name*]# key-life-timekilobytes** <kilobytes>	Define a new maximum lifetime of the security associations in kilobytes.

Creating an ISAKMP transform profile

You need to create at least one isakmp transform profile to define which cryptographic transforms should be used to protect the negotiation of IPSEC security association and the mutual authentication of the IPSEC peers. The parameters defined in this profile are used for the negotiation of ISAKMP security associations in main mode.

The following commands are used to create and configure an ISAKMP transform profile:

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)# profile isakmp-transform** <name>	Create the transform profile with the specified name and enter its configuration mode.
2	***node*(pf-ikptr)**[<*name*>]# **authentica-tionalgorithm md5|sha1**	Define the authentication algorithm to be used, either md5 or sha1.
3	***node*(pf-ikptr)**[<*name*>]# encryption des-cbc|3des-cbc|aes-cbc [key-length]	Define the encryption algorithm and, optionally, the length of the encryption keys in bits to be used.
4 optional	***node*(pf-ikptr)**[<*name*>]# **key-lifetime-seconds <seconds>**	Optionally, you can also change the default ISAKMP security association lifetime in seconds. The default lifetime is 1 day.
5 optional	***node*(pf-ikptr)**[<*name*>]# **key-lifetime-sessions <sessions>**	Optionally, you can also change the default ISAKMP security association lifetime in sessions. This is the maximum number of quick modes that can be created by the ISAKMP SA. By default there is no limit on the number of sessions.

Creating an ISAKMP IPSEC policy profile

You need to create an ISAKMP IPSEC policy profile to define all the settings and profiles needed to establish an IPSEC security association. You can specify the ISAKMP and IPSEC transforms you created above which should be used. You can specify later an ACL with the type of traffic to be treated by a specific ISAKMP IPSEC policy.

The following commands are used to create and configure an ISAKMP IPSEC policy profile:

Mode: Configure

Step	Command	Purpose
1	***node***(cfg)# **profile ipsec-policy-isakmp <name>**	Create the policy profile with a unique name and enter its configuration mode.
2	***node*(pf- ipsik)**[<*name*>]# **authentication-method pre-shared-key <key>**	Define the pre-shared key for authenticating the peers. The key can be a character string of any length.
3	***node*(pf- ipsik)**[<*name*>]# **diffie-hellmangroup {group1|group2|group5}**	Define the diffie-hellman group to be used. **Note:** The higher the group number is, the longer the key length during the diffie-hellman exchange which increases the processing time for the establishment of the shared secret. Especially Group 5 requires a considerable amount of time for processing. You should not use this group in time critical applications unless you know that the tunnel will always be established.
4	***node*(pf- ipsik)**[<*name*>]# **use profile isakmp-transform <name>**	Define one or more ISAKMP transform profiles to be used by this policy. If more than one is defined, IKE will negotiate a transform set supported by both peers.
5	***node*(pf- ipsik)**[<*name*>]# **use profile ipsec-transform <name>**	Define one or more IPSEC transform profiles to be used by this policy. If more than one is defined, IKE will negotiate a transform set, which is supported by both peers.
6	***node*(pf- ipsik)**[<*name*>]# **mode transport|tunnel**	Define the IPSEC encapsulation mode to be used by this policy.
7 optional	***node*(pf- ipsik)**[<*name*>]# **peer <ip or FQDN>**	Optionally define the peer for which this policy should be used. If this policy shall be used for multiple peers in transport mode, do NOT specify a peer. The peer can either be an IP address or a fully qualified domain name.
8 optional	***node*(pf- ipsik)**[<*name*>]# **protectednetwork {host <local-host-ip>}|{subnet <local-subnet-address> <local-subnetmask>}|{range <local-range-start><local-range-end>} {host <remote-hostip>}|{subnet <remote-subnet-address><remote-subnet-mask>}|{range<remote-range-start><remote-rangeend>}**	Optionally if the remote system requires protected networks to be specified in the identity payload of the quick mode, you can define one or more protected networks using this command.

Step	Command	Purpose
9 optional	***node*(pf- ipsik)**[<*name*>]# **protection-group <group>**	If required, you can specify a protection group. The protection-group is a proprietary feature and is not compatible with third-party devices. Therefore do not configure it for connections to third party devices.

Creating/modifying an outgoing ACL profile for IPSEC

This is basically the same as for manual keyed IPSEC connections ("Creating an IPsec policy profile" on page 290). Make sure that your ACL allows traffic from and to UDP port 500 in plaintext to allow ISAKMP messages to be exchanged.

Configuration of an IP interface and the IP router for IPSEC

This is exactly the same as for manual keyed IPSEC connections ("Creating an IPsec policy profile" on page 290).

Policy matching

Normally, if an initial ISAKMP message is received from the network, the system tries to find the corresponding ISAKMP IPSEC policy by matching the received source-ip address with the peer IP address of an IPSEC policy.

However, in applications with dynamic IP addressing, a Fully Qualified Domain Name (FQDN) might be specified as the peer, rather than using an IP address. In this case, it is not possible to find the correct policy using the source-ip address. To solve this problem, you specify the same protection-group ID in the ISAKMP IPSEC policy profiles for all of the peers. The peers should all use the same remote policy. In this case, if the system receives an initial IKE packet, it will search for an ISAKMP IPSEC policy profile, which has the same protection-group ID as the policy that created the ISAKMP packet.

Sample configuration snippet

Below you see a sample of the minimal required settings to be added to a configuration file in order to establish an IKE IPSEC connection:

```
profile acl WAN_Out
  permit 1 esp any any
  permit 2 ah any any
  permit 3 udp any any eq 500
  permit 4 ip any 10.0.0.0 0.255.255.255 ipsec-policy VPN
  permit 5 ip any any


profile ipsec-transform IPSEC_3DES_192
  esp-encryption 3des-cbc 192


profile isakmp-transform ISAKMP_3DES_192
  encryption 3des-cbc 192
  authentication-algorithm sha1
```

```
profile ipsec-policy-isakmp VPN
  authentication-method pre-shared-key sdfkl@hgdslkfs/iuçkfld$gus+ghf
  mode tunnel
  peer 1.2.3.4
  diffie-hellman-group group2
  use profile ipsec-transform 1 IPSEC_3DES_192
  use profile isakmp-transform 1 ISAKMP_3DES_192


context ip
  interface WAN
  use profile acl WAN_Out out
```

Troubleshooting

To analyze IKe configuration or networking problems, use the following debug monitors that log important information about the exchanged ISAKMP messages:

- **debug ike event**

This monitor prints every ISAKMP message sent or received as well as the current state of the ISAKMP main and quick modes.

- **debug ike error**

This monitor prints information about errors detected during the ISAKMP exchange. In addition to the monitors there are also show commands, which display current information about IKE and IPSEC.

- **show ike policy <policy-name>**

Displays information about the configuration options of specific policy as well as an indication, if the policy is valid or not. A policy might be invalid, if one or more configuration option is missing.

- **show ike status**

Displays information about the state of current IKE main and quick modes.

- **show ipsec security-associations**

Displays information about currently established IPSEC security associations including SPIs, peer IP addresses and security association lifetime.

Using an alternate source IP address for specific destinations

Normally, locally originated IP packets use the IP address of the outbound IP interface as their source address. However, when using VPN tunnels there are situations, where locally originated IP packets must be sent using the source IP address of an alternate interface. You can specify using the following command that for one or more destination network the IP address of an alternate IP interface should be used. This configuration command affects all locally originated IP packets except those, which originate from explicitly bound components.

Mode: context ip

Step	Command	Purpose
1	***node*(ctx-ip)**[ctx-*name*]# **[no] sourceaddress-map <destination-network><destination-mask><ip-interface-name>**	Defines that locally originated packets destined for the specified destination network shall use the IP address of the specified IP interface as their source address.

Sample configurations

The following sample configurations establish IPsec connections between an IPLink and a Cisco router. To interconnect two IPLink devices instead, derive the configuration for the second IPLink 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

IPLink 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

IPLink 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
!
...
```

For the remainder of the configuration (see above), just change the name of the IPsec policy profile in the ACL profile *VPN_Out*

IPsec tunnel, 3DES encryption at 192 bit key length, ESP authentication with HMAC-MD5-96

IPLink configuration

```
profile ipsec-transform TDES_MD5
  esp-encryption 3des-cbc 192
  esp-authentication hmac-md5-96

profile ipsec-policy-manual VPN_TDES_MD5
  use profile ipsec-transform TDES_MD5
  session-key inbound esp-authentication 1234567890ABCDEF1234567890ABCDEF
  session-key outbound esp-authentication FEDCBA0987654321FEDCBA0987654321
  session-key inbound esp-encryption
1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF
  session-key outbound esp-encryption
FEDCBA0987654321FEDCBA0987654321FEDCBA0987654321
  spi inbound esp 7777
  spi outbound esp 8888
  peer 200.200.200.1
  mode tunnel
...
```

For the remainder of the configuration (see above), just change the name of the IPsec policy profile in the ACL profile *VPN_Out*

Cisco router configuration

```
crypto ipsec transform-set 3DES_MD5 esp-3des esp-md5-hmac
!
crypto map VPN_3DES_MD5 local-address FastEthernet0/1
crypto map VPN_3DES_MD5 10 ipsec-manual
 set peer 200.200.200.2
 set session-key inbound esp 8888 cipher
FEDCBA0987654321FEDCBA0987654321FEDCBA0987654321 authenticator
FEDCBA0987654321FEDCBA0987654321
 set session-key outbound esp 7777 cipher
1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF authenticator
1234567890ABCDEF1234567890ABCDEF
 set transform-set 3DES_MD5
 match address 110
!
...
```

For the remainder of the configuration (see above), just change the name of the IPsec policy profile in the ACL profile *VPN_Out*.

Appendix A **Terms and definitions**

Chapter contents

Introduction

This chapter contains the terms and their definitions that are used throughout this *IPLink software Software Configuration Guide*. This guide contains many terms that are related to specific networking technologies areas such as LAN protocols, WAN technologies, routing, Ethernet, and Frame Relay. Moreover various terms are related to telecommunication areas, such as the integrated services digital network (ISDN), public switched telephone network (PSTN), and plain old telephone service (POTS).

IPLink software architecture terms and definitions

Term or Definition	Meaning
Administrator	The person who has privileged access to the IPLink software CLI.
Application Download	A application image is downloaded from a remote TFTP server to the persistent memory (flash:) of an IPLink.
Application Image	The binary code of the IPLink software stored in the persistent memory (flash:) of an IPLink.
Batchfile	Script file containing instructions to download one or more software component from a TFTP server to the persistent memory (flash: or nvram:) of an IPLink.
Bootloader	The bootloader is a "mini" application performing basic system checks and starting the IPLink software application. The bootloader also provides minimal network services allowing the IPLink to be accessed and upgraded over the network even if the IPLink software 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 an IPLink.
Bootstrap	The starting-up of an IPLink, 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*.
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.
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*.

Term or Definition	Meaning
Command Line Interface	An interface that allows the user to interact with the IPLink software 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 an IPLink.
Configuration File	The configuration file contains the IPLink software CLI commands, which are used to configure the software modules of the IPLink software performing a certain functionality of the IPLink.
Configuration Server	A central server used as a store for configuration files, which are downloaded to or uploaded from an IPLink using TFTP.
Configuration Upload	A configuration file is uploaded from the persistent memory (nvram:) or volatile memory (system:) of an IPLink via TFTP to a TFTP server.
Context	An IPLink software 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.
Driver Software Download	A driver software image is downloaded from a remote TFTP server to the persistent memory (flash:) of an IPLink.
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 an IPLink.
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 an IPLink.
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 an IPLink containing the Application Image, Bootloader Image and the driver software Image.
flash:	A region in the persistent memory of an IPLink. See also *flash memory.*
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.
Interface	In IPLink software 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 IPLink to the outside world.
Jitter	Jitter is the variation on packets arriving on an IPLink. See also *dejitter buffer.*

Term or Definition	Meaning
Mode	The IPLink software CLI is comprised of modes. There are two basic mode groups, the execution mode group and the configuration mode group.
Network Management System	System responsible for managing at least part of a network. An NMS is generally a reasonably powerful and well-equipped computer, such as an engineering workstation. NMSs communicate with agents to help keep track of network statistics and resources.
Node	Endpoint of a network connection or a junction common to two or more lines in a network. A Node can be a router, e.g. an IPLink. 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 an IPLink or network element.
nvram:	Persistent memory section of an IPLink containing the startup configuration, the factory configuration and used defined configurations.
Operator	The person who has limited access to the IPLink software 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 IPLink 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 IPLink devices. The PMC drivers are interface card specific and also have build numbers. Refer to the IPLink software release notes for PMC driver software compatibility. The PMC drivers may be upgraded together with the IPLink software release or they can be downloaded individually into the persistent memory (flash:) of an IPLink.
PMC Loader	The PMC loader initializes the PMC interface card mounted in IPLink series of devices. It checks hardware versions and determines if compatible PMC drivers are available. The PMC loader may be upgraded together with the IPLink software release.
Port	In IPLink software a port represents a physical connector on the IPLink.
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 that can be used on multiple contexts, interfaces or gateways.
Release	IPLink software is organized in releases that define the main voice and data features of an IPLink. Several builds can be available from certain release. See also *build*.

Term or Definition	Meaning
Routing Engine	In IPLink software the routing engine handles the basic IP routing.
Running Configuration	The currently running configuration (running-config) for IPLink software, which is executed from the volatile memory (system:) on the IPLink.
IPLink software	IPLink software is the application software running on the IPLink hardware platforms. The IPLink software is available in several releases that in general support all currently available IPLink models.
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.
System Image	A collective term for application images and interface card driver software, excluding configuration files.
System Memory	The volatile memory, that includes the system: region, holding the running-config for the IPLink software during operation of an IPLink.
system:	A region in the volatile memory of an IPLink. 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 Mode summary

Chapter contents

Introduction

Figure 42 on page 310 and figure 43 on page 311 show the configuration mode hierarchy. Each box contains the mode name, the command to enter in this mode and the mode prompt printed in a Telnet or console session. The commands are defined in appendix C, "Command summary" on page 313.

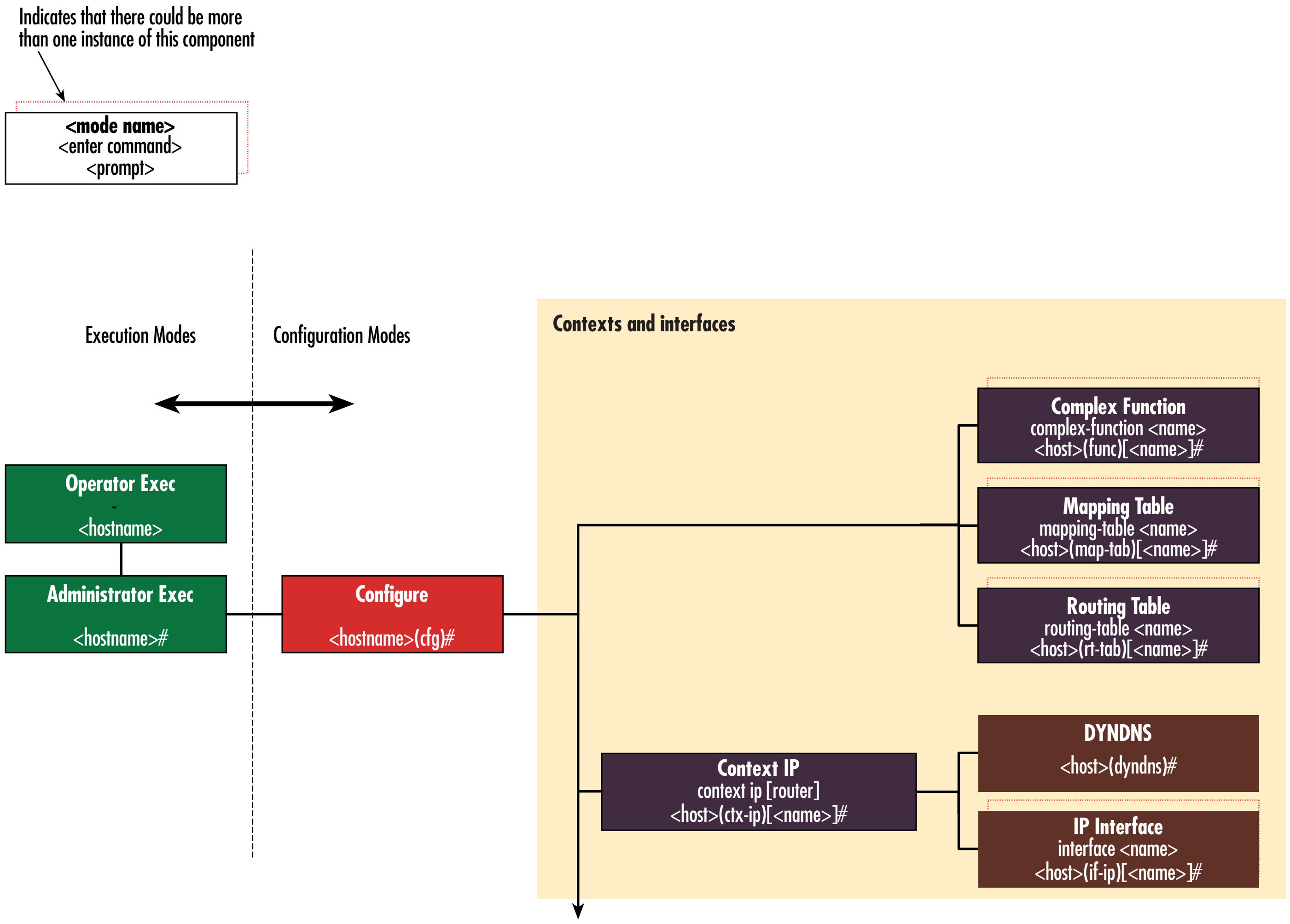


Figure 42. Mode overview, 1 of 2

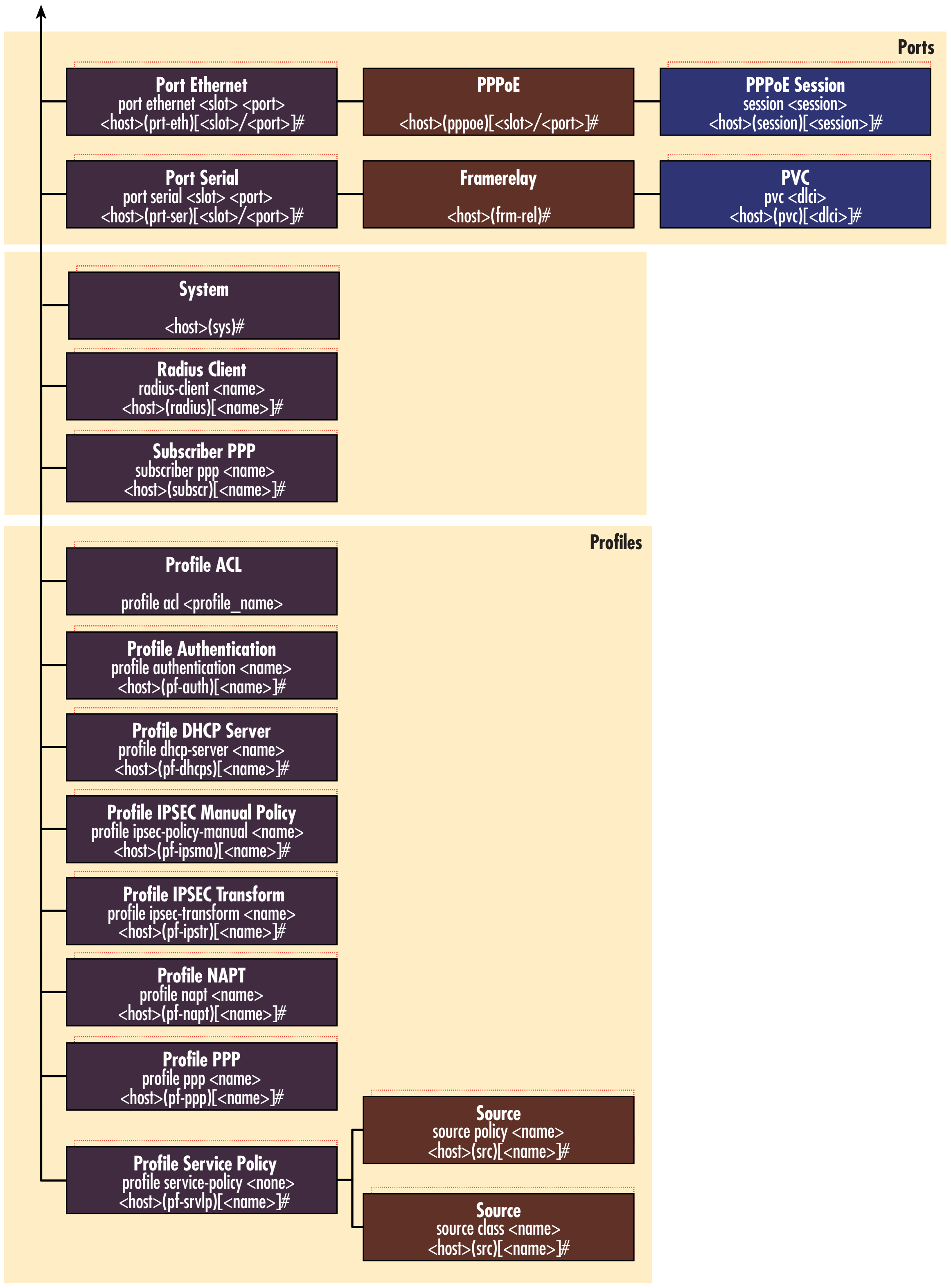


Figure 43. Mode overview, 2 of 2

Appendix C **Command summary**

Chapter contents

Introduction

This command summary is valid for IPLink software Release 3.20. Commands in future IPLink software releases may be different. The information provided in this chapter is subject to change without notice. The command summary is organized as follows:

```
Mode Name
Enter Command
Command 1
…
Exit

Mode Name
…
```

Several commands contain a lot of parameters and arguments. The command syntax is Extended Backus-Naur Form (EBNF) and is described as follows:

- Arguments where you must supply the value are surrounded by <angle brackets>.
- Optional arguments within commands are shown in square brackets ([ ])
- Alternative parameters within commands are separated by vertical bars (|).
- Alternative but required parameters are shown within grouped braces ({ }) and are separated by vertical bars (|).

Command syntax is illustrated by an example in figure 44.

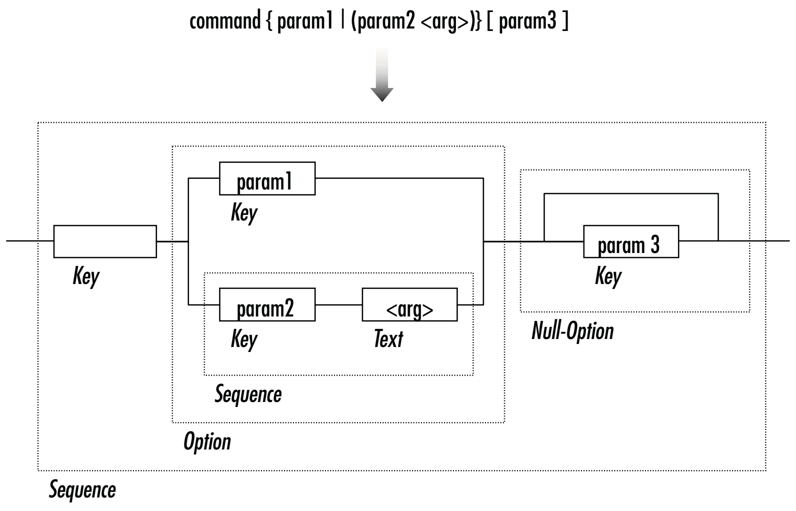


Figure 44. EBNF syntax

Summary

operator_exec

```
operator_exec
   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 ]
   clear
   show clock
   show uptime
   show timer []
   [no] debug timer
   [no] debug arp
   show arp [<interface_show> ]
   arp flush
            <interface_flush>
            <remote-ip>

   show ip route
   show ip interface [<interface_name> ] [router ]
   show napt interface <ip_interface_name_show> [resolve ] [router ]
   show rip [interface <ip_interface_name_show> [router ] ]
   show port ethernet [<print-slot> <print-port> ]
   show port serial [<print-slot> <print-port> ] [detail <detail> ]
   show framerelay [pvc <print-dlci> ]
   show ppp {links|networks} [detail <level> ]
   show pppoe [detail <level> ]
   show log [{event|reset|boot} ]
   show log login
   show service-policy [interface <interface-name> [router ] ]
   show version
   show licenses
   install license <license>
   jobs
   fg <job>
   terminal width {<value> | default }
   terminal height {<value> | default }
   show terminal [<variable-name> ]
   [no] terminal more
   [no] terminal {idle-time-logout|absolute-time-logout} <value>
   [no] terminal mark <value> [{bold|underscore|blink|reverse} ]
   [no] terminal monitor-filter <filter>
   logout
   su <account>
   who
   help
   show history
   show power-management [time <time> ]
```

administrator_exec

```
administrator_exec
```

```
enable
copy {{running-config|factory-config|startup-config|system:running-config} |
   {cli:|preferences:} | <src> | <src> } {{running-config|startup-
   config|system:running-config|flash:|licenses:} | {cli:|preferences:} | <dest> |
   <dest> }
erase {{startup-config} | {cli:|preferences:} | <config> }
edit <file>
debug all
dyndns reset
show dyndns
show dns-client
show crypto offload
[no] debug dyndns
[no] debug linkscheduler
[no] debug acl [{in|out} [detail <detail> ] ]
[no] debug rtm
[no] debug rip
[no] debug sntp-client
[no] debug dhcp-server
show dhcp-server
[no] debug dhcp-client
show dhcp-client
[no] debug ppp [{all|tx-hdlc|rx-hdlc|tx-control|rx-control|tx-option|rx-
   option|timer|state-machine|control|management|authentication|error} ]
[no] debug pppoe [{all|tx-discovery|rx-discovery|tx-session|rx-
   session|timer|state-machine|control|management|error} ]
[no] debug serial
[no] debug framerelay [{all|error|lmi|packet|management} ]
[no] debug ipsec
[no] debug flashserver
show accounts
show {nvram: | {running-config|factory-config|startup-config|system:running-
   config} | {cli:|preferences:} | <config> }
show crc {{running-config|factory-config|startup-config|system:running-config} |
   {cli:|preferences:} | <config> }
show profile napt [<napt-profile_name_show> ]
show subscriber ppp [<subscriber_ppp_name_show> ]
show profile ppp [<profile_ppp_name_show> ]
show ipsec security-associations
show ip ports
show profile ipsec-policy-manual
show profile ipsec-transform
show profile service-policy [<arbiter-name> ]
show snmp
show sntp-client
show provisioning
[no] debug provisioning
provisioning execute <profile> [forced ]
[no] provisioning reset <profile>
show profile acl [<acl_name> ]
[no] debug aaa
show profile authentication [<profile_show> ]
show radius-client [<radius-client_show> ]
[no] debug tftp-client
reload [graceful if-needed forced ]
```

```
show log supervisor
show memory stat
end
```

configure

```
configure
cli version <version>
cli config indention {none|commands-only|modes-and-commands}
[no] cli config defaults
terminal {console|telnet|http} use authentication <profile>
[no] terminal telnet
terminal telnet port <port>
[no] administrator <account> password <password>
[no] operator <account> password <password>
[no] banner <banner>
clock set <time>
[no] dns-client enabled
{dns | dns-client } cache <num_cachesize>
dns-client cache-max-age <age>
[no] {dns-relay | (dns relay ) }
[no] webserver
              (port <port> )
              (language {en|de} )

[no] snmp community <community> {ro|rw}
[no] snmp target <ipAddress> security-name <community>
[no] snmp host <ipAddress> security-name <community>
[no] echo-server [port <echo_port> ]
[no] sntp-client
sntp-client server primary <server_address>
              (port <sntp_port> )
              (version <version_number> )

sntp-client server 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
[no] timer [volatile ] [{now | ([{<from-time-hour-minutes> | (<from-time-
   quantity> minutes ) | ({noon|teatime|midnight} ) } ] [[<from-date-month> ]
   [<from-date-day> ] [<from-date-year> ] ] ) } [{+|-} {(<from-shift-quantity>
   {{minute|hour|day|week|month|year} |
   {minutes|hours|days|weeks|months|years} } ) | (<from-shift-quantity>
   {{minutes|hours|days|weeks|months|years} |
   {minute|hour|day|week|month|year} } ) | (<from-shift-hour-minutes> ) }
   {next|previous} <from-shift-dow> ] ] [until {now | ([{<to-time-hour-minutes>
   | (<to-time-quantity> minutes ) | ({noon|teatime|midnight} ) } ] [[<to-date-
```

```
     month> ] [<to-date-day> ] [<to-date-year> ] ] ) } [{+|-} {(<to-shift-
     quantity> {{minute|hour|day|week|month|year} |
     {minutes|hours|days|weeks|months|years} } ) | (<to-shift-quantity>
     {{minutes|hours|days|weeks|months|years} |
     {minute|hour|day|week|month|year} } ) | (<to-shift-hour-minutes> ) }
     {next|previous} <from-shift-dow> ] ] [every
     {{minute|hour|day|week|month|year} | (<repeat-quantity>
     {{minutes|hours|days|weeks|months|years} |
     {minute|hour|day|week|month|year} } ) |
     {minutes|hours|days|weeks|months|years} } ] [<command-line> ]
  system hostname <string>
  system location <string>
  system contact <string>
  system supplier <string>
  system provider <string>
  system subscriber <string>
```

radius-client

```
  radius-client
     [no] radius-client <radius-client>
     radius-server <hostname>
     [no] shared-secret {authentication} <shared-secret>
     exit


     system

     system
     [no] bypass-mode
     clock-source {internal | ( <slot> <port> ) }
     [no] hairpinning
```

profile_acl

```
  profile_acl
     [no] profile acl <profile_name>
     [no] {permit|deny} {(<index> {({before|after} <new-index> ) | ({up|down}
        <positions> ) } ) | ([[{before|after} ] <index> ] {({ip|ah|esp|gre|igmp}
        {any | (host <src-host-address> ) | ( <src-base-address> <src-wildcard> )
        } {any | (host <dst-host-address> ) | ( <dst-base-address> <dst-wildcard>
        ) } ) | ({tcp|udp|sctp} {any | (host <src-host-address> ) | ( <src-base-
        address> <src-wildcard> ) } { | (eq <src-port-eq> ) | (lt <src-port-lt> )
        | (gt <src-port-gt> ) | (range <src-port-from> <src-port-to> ) } {any |
        (host <dst-host-address> ) | ( <dst-base-address> <dst-wildcard> ) } { |
        (eq <dst-port-eq> ) | (lt <dst-port-lt> ) | (gt <dst-port-gt> ) | (range
        <dst-port-from> <dst-port-to> ) } ) | (icmp {any | (host <src-host-
        address> ) | ( <src-base-address> <src-wildcard> ) } {any | (host <dst-
        host-address> ) | ( <dst-base-address> <dst-wildcard> ) } [name
        {administratively-prohibited|alternate-address|conversion-error|dod-host-
        prohibited|dod-net-prohibited|echo|echo-reply|general-parameter-
        problem|host-isolated|host-precedence-unreachable|host-redirect|host-tos-
        redirect|host-tos-unreachable|host-unknown|host-unreachable|information-
        reply|information-request|mask-reply|mask-request|mobile-redirect|net-
        redirect|net-tos-redirect|net-tos-unreachable|net-unreachable|network-
        unknown|no-room-for-option|option-missing|packet-too-big|parameter-
        problem|port-unreachable|precedence-unreachable|protocol-
        unreachable|reassembly-timeout|redirect|router-advertisement|router-
        solicitation|source-quench|source-route-failed|time-exceeded|timestamp-
        reply|timestamp-request|traceroute|ttl-exceeded|unreachable} type <icmp-
        type> [code <icmp-code> ] ] ) } [dscp <dscp-value> [mask {1|3|7|15|31} ]
        (precedence {{critical|flash|flash-
        override|immediate|internet|network|priority|routine} | <precedence-
        value> } ) [tos {{max-reliability|max-throughput|min-delay|min-monetary-
        cost|normal} | <tos-value> } ] ] [{cos | traffic-class } {<traffic-class1>
        | <new-traffic-class1> } {cos-rtp | rtp-traffic-class } {<traffic-class1>
        | <new-traffic-class1> } {<traffic-class2> | <new-traffic-class2> } ]
        [ipsec-policy <policy> ] ) }
    exit
```

profile_service-policy

```
profile_service-policy
    [no] profile service-policy <arbiter-name>
    mode {shaper | wfq | burst-shaper | burst-wfq }
    [no] rate-limit <rate_limit>
                    (header-length <header_length> )
                    atm-modem
                    (voice-margin <voice_margin> )

    [no] map packet-size {({routed-voice | routed-voice-encrypted } traffic-
        class <traffic-class> ) | (<lower-size> <upper-size> traffic-class
        <traffic-class> ) }
    [no] queue-limit <queue_limit>
    [no] set ip dscp <tag_dscp>
    [no] set ip precedence <tag_prec>
    [no] set ip tos <tag_tos>
    [no] set layer2 cos <tag_l2cos>
    [no] debug queue statistics [detail <debug_queue> ]
```

profile_napt

```
profile_napt
    [no] profile napt <profile>
    [no] range <local_ip1> <local_ip2> <global_ip> [<global_ip2> ]
    [no] static <local_ip1> <global_ip>
    [no] static {ah|esp|gre|ipv6} <local_ip1> [<global_ip> ]
    [no] dmz-host <local_ip1> [<global_ip> ]
    tcp-port-range <start> <end>
    [no] preserve-tcp-ports
    udp-port-range <start> <end>
    [no] preserve-udp-ports
    udp-handling {full-cone|address-restricted-cone|port-restricted-
        cone|symmetric}
    [no] static {udp|tcp} <local_ip1> <local_port> [<global_ip> ] [<global_port>
        ]
    [no] static {udp|tcp} <local_port> <local_ip1>
    exit
```

profile_ppp

```
profile_ppp
    [no] profile ppp <profile_ppp_name>
```

```
   lcp-configure-request interval <interval> max <max>
   lcp-configure-nak max <max>
   lcp-terminate-request interval <interval> max <max>
   lcp-echo-request interval <interval> max <max>
   mtu min <min> max <max> [ignore-link ]
   mru min <min> max <max> [ignore-link ]
   accm <value>
   [no] authentication {(chap pap ) | {chap|pap} } interval <interval> max
      <max>
   [no] callback {active|passive|both} interval <interval> max <max>
   ipcp-configure-request interval <interval> max <max>
   ipcp-configure-nak max <max>
   ipcp-terminate-request interval <interval> max <max>
   [no] van-jacobson {compression|decompression} max-slots <max>
   [no] local-address-autoconfig
   exit
```

profile-ipsec-transform

```
profile-ipsec-transform
   [no] profile ipsec-transform <name>
   [no] esp-encryption {(aes-cbc [128 192 256 ] ) | (des-cbc [64 ] ) | (3des-cbc
      [128 192 ] ) | (null ) }
   [no] esp-authentication {hmac-md5-96 | hmac-sha1-96 }
   [no] ah-authentication {hmac-md5-96 | hmac-sha1-96 }
   exit
```

ipsec-manual-policy

```
ipsec-manual-policy
   [no] profile ipsec-policy-manual <name>
   [no] use profile ipsec-transform <ts-name>
   [no] session-key {inbound|outbound} {ah-authentication|esp-
      authentication|esp-encryption} <key>
   [no] spi {inbound|outbound} {ah|esp} <spi>
   [no] peer <peer>
   [no] mode {tunnel|transport}
   exit
```

profile_dhcp-server

```
profile_dhcp-server
   [no] profile dhcp-server <dhcps_profile_name>
   network [[{before|after} ] <index> ] <address> <mask>
   [no] include [[{before|after} ] <index> ] <from> <to>
   lease {( <time> {days|hours|minutes} ) | infinite }
   [no] default-router [[{before|after} ] <index> ] <address>
   [no] domain-name <name>
   [no] domain-name-server [[{before|after} ] <index> ] <address>
   [no] netbios-name-server [[{before|after} ] <index> ] <address>
   [no] netbios-node-type {{b-node|p-node|m-node|h-node} }
   [no] bootfile <bootfile>
   [no] next-server <address>
   exit
```

profile_authentication

```
profile_authentication
   [no] profile authentication <profile>
   [no] method [[{before|after} ] <index> ] {(radius <method-name> ) | local |
      none }
   server-timeout [<timeout> ]
   exit
```

profile_provisioning

```
profile_provisioning
   [no] profile provisioning <profile>
   destination {configuration|script}
   [no] location [<index> ] <location> [encryption-key <encryption_text> ]
   [no] activation reload {immediate|graceful|deferred}
   exit
```

context_ip

```
context_ip
   context ip [router ]
   [no] dhcp-server
   rtp-port-range [<start> <end> ]
   dhcp-server clear-lease {all | <address> }
   [no] dhcp-server use <name>
   [no] route <destaddr> <destmask> {<gwaddr> | <interface> } [<metric> ]
```

interface

```
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
   [no] use profile acl <acl_profile_name> out
   dhcp-client {renew|release}
   [no] {cos | traffic-class } <cos_group>
   [no] use profile service-policy <arbiter-name> in
   [no] use profile service-policy <arbiter-name> 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 <profile>
        [no] napt-inside
        [no] tcp adjust-mss rx {mtu | ( <mss> ) }
        [no] tcp adjust-mss tx {mtu | ( <mss> ) }
        exit
```

dyndns

```
    dyndns
        dyndns
        [no] hostname <host> [wildcard ]
        [no] authentication <user> <passwd>
        [no] service {custom | dynamic | static }
        [no] observe <ifname>
        [no] mail-exchanger <host> [backup-mx ]
        exit
    exit
```

subscriber_ppp

```
subscriber_ppp
    [no] subscriber ppp <subscriber_ppp_name>
    dial [{in|out} ]
    [no] authentication chap
    [no] authentication pap
    [no] authentication chap pap
    [no] identification outbound <id> [password <password> ]
    [no] identification inbound <id> [password <password> ]
    [no] timeout absolute <timeout>
    [no] timeout idle <timeout>
    [no] timeout on-timer
    [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
```

port_ethernet

```
port_ethernet
    port ethernet <slot> <port>
    medium {auto | ( {10|100} {half|full} ) }
    encapsulation {ip|pppoe|multi}
    [no] bind interface <interface> [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
```

vlan

```
      vlan

         [no] vlan <vlan>
         [no] map cos <cos> to <traffic-class>
         encapsulation {ip|pppoe|multi}
         [no] bind interface <interface> [router ]
         [no] shutdown
         exit
      exit
```

port_serial

```
   port_serial

      port serial <slot> <port>
      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] baudrate [<baudrate> ]
      [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
         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
```

Other

Show help

Step	Command	Purpose
1	**help** [*topic*]	Shows command help.

Show command history

Step	Command	Purpose
1	**show history**	Shows command history.

Use CTRL-N and CTRL-P to browse. The cursor keys (up, down) are not working.

Show RedBoot version

Step	Command	Purpose
1	**show version**	Shows RedBoot version.

Restart system

Step	Command	Purpose
1	**reload**	Restarts the system.

Check network connection to remote system

Step	Command	Purpose
1	**ping** *IP_address*	Check network connection to a remote system, by sending ICMP echo requests and listening for ICMP echo replies.

Appendix D **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	IPLink software
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
IPLink	IPLink
ISDN	Integrated Services Digital Network
ISDN NT	ISDN Network Termination
ISDN S	ISDN S(ubscriber Line) Interface
ISDN T	ISDN T(runk Line) Interface
ISDN TE	ISDN Network Terminal Mode
ITC	Information Transfer Bearer Capability
L	
L2TP	Layer Two Tunneling Protocol
LAN	Local Area Network
LCR	Least Cost Routing
LDAP	Lightweight Directory Access Protocol
LE	Local Exchange
LED	Light Emitting Diode
LT	Line Termination

Abbreviation	Meaning
M	
MGCP	Media Gateway Control Protocol
MIB II	Management Information Base II
Modem	Modulator – Demodulator
MSN	Multiple Subscriber Number
N	
NAPT	Network Address Port Translation
NAT	Network Address Translation
NIC	Network Interface Card
NT	Network Termination
NT1	Network Termination 1
NT2	Network Termination 2
NT2ab	Network Termination with 2a/b Connections
O	
OEM	Original Equipment Manufacturer
OSF	Open Software Foundation
OSPF	Open Shortest Path First
P	
PBR	Policy Based Routing (principles)
PBX	Private Branch Exchange
PC	Personal Computer
PMC	Production Technology Management 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
RTM	Route Table Manager

Abbreviation	Meaning
RTP	Real-time Protocol
S	
S1	IPLink-connection for Trunk Line
S2	IPLink-connection for Subscriber Line
SAR	Segmentation and Reassembly
S-Bus	Subscriber Line (Connection) Bus
SCN	Switched Circuit Network
SCTP	Stream Control Transmission Protocol
SDSL	Symmetric Digital Subscriber Line
SGCP	Simple Gateway Control Protocol
SIP	Session Initiation Protocol.
SME	Small and Medium Enterprises
SNMP	Simple Network Management Protocol
SOHO	Small Office Home Office
SONET	Synchronous Optical Network
SS7	Signaling System No. 7
STM	SDH Transmission at 155 Mb/s
SVC	Switched Virtual Circuit
SW	Software
T	
TCP/IP	Transport Control Protocol/Internet Protocol
TE	Terminal Equipment
TFTP	Trivial File Transfer Protocol
U	
UBR	Unspecified Bit Rate
UD 64	Unrestricted Data 64 kb/s
UDP	User Datagram Protocol
V	
VBR	Variable Bit Rate
VCI	Virtual Channel Identifier
VoIP	Voice over Internet Protocol
VPI	Virtual Path Identifier
W	
WAN	Wide Area Network

Appendix E Used IP ports in the IPLink software

Chapter contents

Used IP ports in the IPLink software

Component	Port	Description
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 IPLink)