PATTON Electronics Co.

SmartNode Series
SmartWare Release 5.2

Software Configuration Guide

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

Part Number: **07MSWR52_SCG, Rev. A**
Revised: **August 7, 2008**

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

Copyright Statement

Copyright © 2008, Patton Electronics Company. All rights reserved.

Trademark Statement

The terms *SmartWare, SmartView, SmartLink,* and *SmartNode* are trademarks of Patton Electronics Company. All other trademarks presented in this document are the property of their respective owners.

Notices

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

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

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

Supported Platforms

SmartNode 4110 Series	SmartNode 4830 Series	SmartNode 4552, 4562
SmartNode 4520 Series	SmartNode 4900 Series	SmartNode 4554, 4564
SmartNode 4600 Series	SmartNode 4960	Smart-DTA

Summary Table of Contents

1 System overview 38
2 Configuration concepts 44
3 Command line interface (CLI) 49
4 Accessing the CLI 53
5 System image handling 65
6 Configuration file handling 76
7 Basic system management 90
8 RADIUS Client Configuration 102
9 IP context overview 114
10 IP interface configuration 120
11 NAT/NAPT configuration 132
12 Ethernet port configuration 141
13 Link scheduler configuration 151
14 Serial port configuration 170
15 Frame Relay configuration 177
16 PRI port configuration 191
17 BRI port configuration 205
18 ISDN Overview 212
19 ISDN configuration 217
20 RBS configuration 225
21 DSL Port Configuration 230
22 Basic IP routing configuration 235
23 RIP configuration 242
24 Access control list configuration 253
25 SNMP configuration 267
26 SNTP client configuration 282
27 DHCP configuration 292
28 DNS configuration 304
29 DynDNS configuration 308
30 PPP configuration 313
31 CS context overview 339

32 VPN configuration 362
33 CS interface configuration 381
34 ISDN interface configuration 390
35 FXS interface configuration 404
36 FXO interface configuration 414
37 RBS interface configuration 426
38 H.323 interface configuration 431
39 SIP interface configuration 441
40 Call router configuration 456
41 SIP call-router services 524
42 Tone configuration 529
43 FXS port configuration 537
44 FXO port configuration 542
45 H.323 gateway configuration 546
46 Context SIP gateway overview 559
47 VoIP profile configuration 573
48 PSTN profile configuration 597
49 SIP profile configuration 601
50 Authentication Service 604
51 Location Service 607
52 VoIP debugging 624
A Terms and definitions 644
B Mode summary 650
C Command summary 654
D Internetworking terms & acronyms 657
E Used IP ports & available voice codecs 662

Table of Contents

Summary Table of Contents **3**
Table of Contents **5**
List of Figures **27**
List of Tables **29**
About this guide **30**
Audience 30
How to read this guide 30
Structure 31
Precautions 34
Typographical conventions used in this document 34
General conventions 34
Mouse conventions 36
Service and support 36
Patton support headquarters in the USA 36
Alternate Patton support for Europe, Middle East, and Africa (EMEA) 36
Warranty Service and Returned Merchandise Authorizations (RMAs) 37
Warranty coverage 37
Returns for credit 37
Return for credit policy 37
RMA numbers 37
Shipping instructions 37
1 System overview **38**
Introduction 39
SmartWare embedded software 40
Applications 41
Carrier networks 41
Enterprise networks 42
LAN telephony 43
2 Configuration concepts **44**
Introduction 45
Contexts and Gateways 46
Context 46
Gateway 46
Interfaces, Ports, and Bindings 47
Interfaces 47
Ports and circuits 47
Bindings 47
Profiles and Use commands 48
Profiles 48
Use Commands 48

3 Command line interface (CLI) 49
Introduction 50
Command modes 50
CLI prompt 50
Navigating the CLI 51
Initial mode 51
System changes 51
Configuration 51
Changing Modes 51
Command editing 51
Command help 51
The No form 51
Command completion 51
Command history 52
Command Editing Shortcuts 52
4 Accessing the CLI 53
Introduction 54
Accessing the SmartWare CLI task list 54
Accessing via the console port 55
Console port procedure 55
Telnet Procedure 56
Using an alternate TCP listening port for the Telnet server 56
Disabling the Telnet server 56
Logging on 56
Selecting a secure password 57
Password encryption 58
Factory preset administrator account 58
Creating an operator account 58
Creating an administrator account 59
Opening a secure configuration session over SSH 59
Displaying the CLI version 60
Displaying account information 60
Switching to another account 61
Checking identity and connected users 61
Command index numbers 62
Ending a Telnet or console port session 64
Showing command default values 64
5 System image handling 65
Introduction 66
Memory regions in SmartWare 67
System image handling task list 68
Displaying system image information 69
Copying system images from a network server to Flash memory 69

Upgrading the software directly 71
Auto provisioning of firmware and configuration 72
Boot procedure 74
Factory configuration 75
Default Startup Configuration 75
IP Addresses in the Factory Configuration 75

6 Configuration file handling 76
Introduction 77
Understanding configuration files 77
Factory configuration 79
Configuration file handling task list 79
Copying configurations within the local memory 80
Replacing the startup configuration with a configuration from Flash memory 81
Copying configurations to and from a remote storage location 82
Replacing the startup configuration with a configuration downloaded from TFTP server 83
Displaying configuration file information 83
Modifying the running configuration at the CLI 84
Modifying the running configuration offline 85
Deleting a specified configuration 86
Encrypted file download 87
Encrypted Configuration Download 87
Use Cases 88

7 Basic system management 90
Introduction 91
Basic system management configuration task list 91
Managing feature license keys 92
Setting system information 93
Setting the system banner 94
Setting time and date 95
Display clock information 95
Display time since last restart 96
Configuring the Web server 96
Determining and defining the active CLI version 96
Restarting the system 97
Displaying the system logs 97
Displaying reports 98
Controlling command execution 98
Timed execution of CLI command 100
Displaying the checksum of a configuration 100
Configuration of terminal sessions 100

8 RADIUS Client Configuration 102
Introduction 103
The AAA component 103

General AAA Configuration 104
RADIUS configuration 106
Configuring RADIUS clients 107
Configuring RADIUS accounting 108
Configuring the RADIUS server 110
Attributes in the RADIUS request message 110
Attributes in the RADIUS accept message 111
Configuring the local database accounts 111
Storing call logs with quality information 113

9 IP context overview 114
Introduction 115
IP context overview configuration task list 116
Planning your IP configuration 117
IP interface related information 117
QoS related information 117
Configuring physical ports 117
Creating and configuring IP interfaces 117
Configuring NAPT 118
Configuring static IP routing 118
Configuring RIP 118
Configuring access control lists 119
Configuring quality of service (QoS) 119

10 IP interface configuration 120
Introduction 121
IP interface configuration task list 121
Creating an IP interface 121
Deleting an IP interface 122
Setting the IP address and netmask 123
Configuring a NAPT DMZ interface 123
ICMP message processing 124
ICMP redirect messages 124
Router advertisement broadcast message 124
Defining the MTU and MSS of the interface 125
Configuring an interface as a point-to-point link 126
Displaying IP interface information 126
Displaying dynamic ARP entries 127
Flushing dynamic ARP entries 127
Processing gratuitous ARP requests 127
Testing connections with the ping command 127
IP link supervision 128
Check connectivity of an IP link 129
Show IP link status 129
Debug connectivity 129

Debug ARP 129
Traceroute 130
Configuring the IGMP Proxy 131
11 NAT/NAPT configuration 132
Introduction 133
Dynamic NAPT 133
Static NAPT 134
Dynamic NAT 134
Static NAT 135
NAPT traversal 135
NAT/NAPT configuration task list 136
Creating a NAPT profile 136
Configuring a NAPT DMZ host 137
Defining NAPT port ranges 137
Preserving TCP/UDP port numbers in NAPT 138
Defining the UDP NAPT type 138
Activate NAT/NAPT 139
Displaying NAT/NAPT configuration information 139
Configuring NAT static protocol entries 140
12 Ethernet port configuration 141
Introduction 142
Ethernet port configuration task list 142
Entering the Ethernet port configuration mode 142
Configuring medium for an Ethernet port 142
Configuring Ethernet encapsulation type for an Ethernet port 143
Binding an Ethernet port to an IP interface 143
Multiple IP addresses on Ethernet ports 144
Configuring a VLAN 145
Configuring layer 2 CoS to service-class mapping for an Ethernet port 146
Adding a receive mapping table entry 147
Adding a transmit mapping table entry 148
Closing an Ethernet port 148
Using the built-in Ethernet sniffer 149
13 Link scheduler configuration 151
Introduction 152
Applying scheduling at the bottleneck 152
Using traffic classes 152
Introduction to Scheduling 153
Priority 153
Weighted fair queuing (WFQ) 153
Shaping 153
Burst tolerant shaping or wfq 154
Hierarchy 154

Quick references 155
Setting the modem rate 155
Command cross reference 156
Link scheduler configuration task list 156
Defining the access control list profile 157
Packet classification 157
Creating an access control list 158
Creating a service policy profile 159
Specifying the handling of traffic-classes 161
Defining fair queuing weight 161
Defining the bit-rate 162
Defining absolute priority 162
Defining the maximum queue length 162
Specifying the type-of-service (TOS) field 162
Specifying the precedence field 163
Specifying differentiated services codepoint (DSCP) marking 163
Specifying layer 2 marking 164
Defining random early detection 165
Discarding Excess Load 165
Quality of Service for routed RTP streams 165
Devoting the service policy profile to an interface 167
Displaying link arbitration status 168
Displaying link scheduling profile information 168
Enable statistics gathering 168

14 Serial port configuration 170
Introduction 171
Serial port configuration task list 171
Disabling an interface 171
Enabling an interface 172
Configuring the serial encapsulation type 173
Configuring the hardware port protocol 173
Configuring the active clock edge 174
Configuring the baudrate 175

15 Frame Relay configuration 177
Introduction 178
Frame Relay configuration task list 178
Configuring Frame Relay encapsulation 178
Configuring the LMI type 179
Configuring the keep-alive interval 179
Enabling fragmentation 180
Entering Frame Relay PVC configuration mode 181
Configuring the PVC encapsulation type 182
Binding the Frame Relay PVC to IP interface 182

Enabling a Frame Relay PVC 184
Disabling a Frame Relay PVC 184
Debugging Frame Relay 185
Displaying Frame Relay information 186
Integrated service access 187
Example 1: Frame Relay on e1t1 without a channel-group 189

16 PRI port configuration 191
Introduction 192
PRI port configuration task list 192
Enable/Disable PRI port 193
Configuring PRI port-type 193
Configuring PRI clock-mode 193
Configuring PRI line-code 193
Configuring PRI framing 194
Configuring PRI line-build-out (E1T1 in T1 mode only) 195
Configuring PRI used-connector (E1T1 in E1 mode only) 195
Configuring PRI application mode (E1T1 only) 195
Configuring PRI LOS threshold (E1T1 only) 196
Configuring PRI Loopback detection (E1T1 only) 196
Configuring PRI encapsulation 197
Create a Channel-Group 198
Configuring Channel-Group Timeslots 198
Configuring Channel-Group Encapsulation 198
Entering HDLC Configuration Mode 199
Configuring HDLC CRC-Type 199
Configuring HDLC Encapsulation 200
PRI Debugging 200
PRI Configuration Examples 201
Example 1: ISDN 202
Example 2: RBS without a channel-group 202
Example 3: RBS with a channel-group 202
Example 4: Frame Relay without a channel-group 203
Example 5: Framerelay with a channel-group 204
Example 6: PPP without a channel-group 204
Example 7: PPP with a channel-group 204

17 BRI port configuration 205
Introduction 206
BRI port configuration task list 206
Enable/Disable BRI port 206
Configuring BRI clock-mode 206
Configuring BRI Power-Feed 207
Configuring BRI encapsulation 207
Creating a channel group 207

Configuring channel-group timeslots 208
Configuring channel-group encapsulation 208
Entering HDLC configuration mode 208
Configuring HDLC encapsulation 208
BRI Debugging 209
BRI Configuration Examples 210
Example 1: ISDN with auto clock/uni-side settings 210
Example 2: ISDN with manual clock/uni-side settings 210
Example 3: Multi-Link PPP over two B-Channels 211

18 ISDN Overview 212
Introduction 213
ISDN reference points 213
Possible SmartNode port configurations 214
ISDN UNI Signaling 214
ISDN Configuration Concept 216
ISDN Layering 216

19 ISDN configuration 217
Introduction 218
ISDN configuration task list 218
Enter Q.921 configuration mode 218
Configuring Q.921 parameters 218
Configuring Q.921 encapsulation 219
Enter Q.931 configuration mode 219
Configuring Q.931 parameters 220
Configuring Q.931 encapsulation 222
Debugging ISDN 222
ISDN Configuration Examples 223

20 RBS configuration 225
Introduction 226
RBS configuration task list 226
Enter RBS configuration mode 226
Configuring RBS protocol 226
Configuring RBS encapsulation 227
Debugging RBS 227
RBS Configuration Examples 228

21 DSL Port Configuration 230
Introduction 231
Line Setup 231
Configuring PPPoE 231
Configuration Summary 232
Setting up permanent virtual circuits (PVC) 233
Using PVC channels in bridged Ethernet mode 233

Using PVC channels with PPPoE 233
Diagnostics 234
Troubleshooting DSL Connections 234
22 Basic IP routing configuration 235
Introduction 236
Routing tables 236
Static routing 236
Policy routing 236
Basic IP routing configuration task list 236
Configuring static IP routes 237
Deleting static IP routes 238
Displaying IP route information 238
Configuring policy routing 239
Examples 240
Basic static IP routing example 240
Changing the default UDP port range for RTP and RTCP 241
23 RIP configuration 242
Introduction 243
Routing protocol 243
RIP configuration task list 244
Enabling send RIP 244
Enabling an interface to receive RIP 245
Specifying the send RIP version 245
Specifying the receive RIP version 246
Enabling RIP learning 246
Enabling an interface to receive RIP 247
Enabling RIP announcing 247
Enabling RIP auto summarization 248
Specifying the default route metric 248
Enabling RIP split-horizon processing 249
Enabling the poison reverse algorithm 249
Enabling holding down aged routes 250
Setting the RIP route expiry 250
Displaying RIP configuration of an IP interface 251
Displaying global RIP information 251
24 Access control list configuration 253
Introduction 254
About access control lists 254
What access lists do 254
Why you should configure access lists 254
When to configure access lists 255
Features of access control lists 255
Access control list configuration task list 256

Mapping out the goals of the access control list 256
Creating an access control list profile and enter configuration mode 257
Adding a filter rule to the current access control list profile 257
Adding an ICMP filter rule to the current access control list profile 259
Adding a TCP, UDP or SCTP filter rule to the current access control list profile 261
Binding and unbinding an access control list profile to an IP interface 263
Displaying an access control list profile 264
Debugging an access control list profile 264
Examples 266
Denying a specific subnet 266

25 SNMP configuration 267
Introduction 268
Simple Network Management Protocol (SNMP) 268
SNMP basic components 268
SNMP basic commands 268
SNMP management information base (MIB) 269
Network management framework 269
Identification of a SmartNode via SNMP 269
SNMP tools 270
SNMP configuration task list 270
Setting basic system information 270
Setting access community information 272
Setting allowed host information 274
Specifying the default SNMP trap target 274
Displaying SNMP related information 275
Using the AdventNet SNMP utilities 275
Using the MibBrowser 276
Using the TrapViewer 277
Standard SNMP version 1 traps 279
SNMP interface traps 280

26 SNTP client configuration 282
Introduction 283
SNTP client configuration task list 283
Selecting SNTP time servers 284
Defining SNTP client operating mode 284
Defining SNTP local UDP port 285
Enabling and disabling the SNTP client 286
Defining SNTP client poll interval 286
Defining SNTP client constant offset to GMT 287
Defining the SNTP client anycast address 287
Enabling and disabling local clock offset compensation 288
Showing SNTP client related information 289
Debugging SNTP client operation 289

Recommended public SNTP time servers....290
NIST Internet time service....290
Additional information on NTP and a list of other NTP servers....291

27 DHCP configuration....292
Introduction....293
DHCP-client configuration tasks....294
Enable DHCP-client on an IP interface....294
Release or renew a DHCP lease manually (advanced)....296
Get debug output from DHCP-client....296
DHCP-server configuration tasks....297
Configure DHCP-server profiles....297
Use DHCP-server profiles and enable the DHCP-server....299
Define the bootfile (Option 67) for the DHCP-server....300
Define the TFTP server (Option 66) for the DHCP-server....300
Check DHCP-server configuration and status....300
Get debug output from the DHCP-server....301
Configure DHCP-relay....302
Create/Modify DHCP-Relay profile....302
Enable/Disable DHCP-Relay Agent....303

28 DNS configuration....304
Introduction....305
DNS configuration task list....305
Enabling the DNS resolver....305
Enabling the DNS relay....306

29 DynDNS configuration....308
Introduction....309
DynDNS configuration task list....309
Creating a DynDNS account....309
Configuring the DNS resolver....309
Configuring basic DynDNS settings....310
Configuring the DynDNS server....310
Configuring advanced DynDNS settings (optional)....311
Defining a mail exchanger for your hostname....311
Troubleshooting....311

30 PPP configuration....313
Introduction....314
PPP configuration task list....315
Creating an IP interface for PPP....315
Disable interface IP address auto-configuration from PPP....317
Creating a PPP subscriber....317
Trigger forced reconnect of PPP sessions using a timer....319
Disable interface IP address auto-configuration from PPP....319

Configuring a PPPoE session 319
Configuring PPP over a HDLC Link 321
Creating a PPP profile 321
Configuring the local and remote PPP MRRU 323
Displaying PPP configuration information 324
Debugging PPP 325
Sample configurations 329
PPP over Ethernet (PPPoE) 329
Without authentication, encapsulation multi, with NAPT 329
With authentication, encapsulation PPPoE 329
PPP over a HDLC Link (Serial Port) 330
Without authentication, numbered interface 330
With authentication, unnumbered interface 330
PPP over a HDLC Link (E1T1 Port) 330
Without authentication, numbered interface 330
PPP Dial-up over ISDN 331
PPP Dialer 331
Create a dialer 332
Create outbound destinations 332
Configure recovery strategy 333
Create inbound destinations 334
Debug dialer functionality 336
Example – Dial-on demand feature 336
Dial-up 337
Dial-up on demand 337
Dial-up on monitor 338
Dial-up nailed 338
....338

31 CS context overview 339
Introduction 340
CS context configuration task list 341
Planning the CS configuration 341
Configuring general CS settings 343
Configuring the clock source 343
Debugging the clock source 344
Selecting PCM law compression 345
Configuring call routing 345
Creating and configuring CS interfaces 346
Specify call routing 346
Configuring dial tones 347
Configuring voice over IP parameters 347
Configuring ISDN ports 348
Configuring FXS ports 348

Configuring an H.323 VoIP connection 348
Configuring a SIP VoIP connection 348
Activating CS context configuration 349
Planning the CS context 352
Configuring general CS settings 353
Configuring call routing 353
Configuring VoIP settings 355
Configuring BRI ports 355
Configuring an H.323 VoIP connection 356
Activating the CS context configuration 356
Showing the running configuration 358

32 VPN configuration 362
Introduction 363
Authentication 363
Encryption 363
Transport and tunnel modes 364
Permanent IKE Tunnels 364
Key management 364
VPN configuration task list 365
Creating an IPsec transformation profile 365
Creating an IPsec policy profile 365
Creating/modifying an outgoing ACL profile for IPsec 367
Configuration of an IP interface and the IP router for IPsec 368
Displaying IPsec configuration information 368
Debugging IPsec 369
Key management (IKE) 370
Main differences between manual & IKE IPSEC configurations 370
Creating an ISAKMP transform profile 371
Creating an ISAKMP IPSEC policy profile 372
Creating/modifying an outgoing ACL profile for IPSEC 373
Configuration of an IP interface and the IP router for IPSEC 373
Policy matching 373
Sample configuration snippet 373
Troubleshooting 374
Encrypted Voice - Performance considerations 375
Performance considerations 375
Enabling RTP encryption support 375
Using an alternate source IP address for specific destinations 376
Sample configurations 377
IPsec tunnel, DES encryption 377
SmartNode configuration 377
Cisco router configuration 378
IPsec tunnel, AES encryption at 256 bit key length, AH authentication with HMAC-SHA1-96 378

SmartNode configuration 378
Cisco router configuration 378
IPsec tunnel, 3DES encryption at 192 bit key length, ESP authentication with HMAC-MD5-96 379
SmartNode configuration 379
Cisco router configuration 379

33 CS interface configuration 381
Introduction 382
CS interface configuration task list 382
Creating and configuring CS interfaces 383
Configuring call routing 384
Configuring the interface mapping tables 385
Configuring the precall service tables 388

34 ISDN interface configuration 390
Introduction 391
ISDN interface configuration task list 391
Configuring DTMF dialing (optional) 392
Configuring an alternate PSTN profile (optional) 392
Configuring ringback tone on ISDN user-side interfaces 393
Configuring call waiting (optional) 393
Disabling call-waiting on ISDN DSS1 network interfaces 393
Configuring Call-Hold on ISDN interfaces 394
Enabling Display Information Elements on ISDN Ports 394
Configuring date/time publishing to terminals (optional) 394
Sending the connected party number (COLP) (optional) 395
Enabling sending of date and time on ISDN DSS1 network interfaces 395
Defining the 'network-type' in ISDN interfaces 395
ISDN Explicit Call Transfer support (& SIP REFER Transmission) 395
ISDN Advice of Charge support 397
ISDN DivertingLegInformation2 Facility 401
Transmit Direction 401
Receive Direction 401
T1 Caller-Name Support 401

35 FXS interface configuration 404
Introduction 405
FXS interface configuration task list 405
Configuring a subscriber number (recommended) 405
Configuring an alternate PSTN profile (optional) 406
Configuring caller-ID presentation (optional) 406
Configuring flash hook processing (optional) 406
Configuring ringing-cadence (optional) 407
Configuring the Message Waiting Indication feature for FXS 408
Configuration 408
Frequency-shift keying 409

FXS supplementary services description 410
Call hold 411
Call waiting 411
Call waiting reminder ring 412
Drop passive call 412
Drop active call 412
Call toggle 412
Call transfer 412
Conferencing 413
Call park 413

36 FXO interface configuration 414
Introduction 415
FXO services description 416
Creating an FXO interface 416
Deleting an FXO interface 417
FXO interface configuration task list 418
FXO off-hook on caller ID 418
Configuring an alternate PSTN profile (optional) 418
Configuring when the digits are dialed (optional) 419
Configuring the number of rings to wait before answering the call (optional) 421
Configuring how to detect a call has disconnected (optional) 422
Configuring how to detect an outgoing call is connected (optional) 423
Configuring the destination of the call 424
FXO Mute dialing 424
FXO interface examples 425

37 RBS interface configuration 426
Introduction 427
RBS interface configuration task list 427
Creating/Deleting a RBS interface 427
Configuring an alternate PSTN profile 427
Configuring an alternate Tone-Set profile 428
Configuring B-Channel allocation strategy 428
Configuring additional disconnect signals 428
Configuring number of Rings before Off-Hook 429
Configuring ready to dial strategy 429
RBS interface debugging 429

38 H.323 interface configuration 431
Introduction 432
H.323 interface configuration task list 432
Binding the interface to an H.323 gateway 433
Configuring an alternate VoIP profile (optional) 434
Configuring CLIP/CLIR support (optional) 435
Enabling ‘early-proceeding’ on H.323 interfaces 436

Enabling the early call connect (optional) 436
Enabling the early call disconnect (optional) 437
Enabling the via address support (optional) 437
Override the default destination call signaling port (Optional) 437
Configuring status inquiry settings (optional) 438
AOC-D Support for H.323 439

39 SIP interface configuration 441
Introduction 442
SIP interface configuration task list 442
Binding the interface to a SIP gateway 443
Configure a remote host 443
Configuring a local host (Optional) 444
Using an alternate VoIP profile (Optional) 444
Using an alternate SIP profile (Optional) 445
Using an alternate Tone-Set profile (Optional) 445
Configuring early call connect / disconnect (Optional) 446
Configuring address translation (Optional) 446
Mapping call-control properties in SIP headers 446
Mapping SIP headers to call-control properties 447
Configuring ISDN Redirecting Number Tunneling Over SIP 447
Enabling SIP RFC Privacy, Asserted-Identity, & Preferred-Identity headers (RFC 3323/3325) 448
Updating caller address parameters 448
SIP Diversion Header 449
Transmit Direction 450
Receive Direction 450
SIP REFER Transmission (& ISDN Explicit Call Transfer support) 451
AOC Over SIP (Optional) 453
Enabling the session timer (Optional) 454
Enabling the SIP penalty-box feature (Optional) 454
Initiating a new SIP session for redirected SIP calls (Optional) 454
Configure the SIP hold method (Optional) 455

40 Call router configuration 456
Introduction 458
Call router configuration task list 460
Map out the goals for the call router 460
Enable advanced call routing on circuit interfaces 461
Configure general call router behavior 461
Configure address completion timeout 461
Configure default digit collection timeout and terminating character 462
Configure number prefix for ISDN number types 463
Configure call routing tables 464
Create a routing table 464
Called party number routing table 466

Regular Expressions 466
Digit Collection 468
Digit Collection Variants 469
Calling party number routing table 472
Number type routing table 472
Numbering plan routing table 473
Name routing table 474
IP address routing table 474
URI routing table 475
Presentation Indicator Routing Table 475
Screening Indicator Routing Table 476
Information transfer capability routing table 477
Call-router support for redirecting number and redirect reason 478
Time of day routing table 479
Day of Week Routing Table 479
Date routing table 479
Deleting routing tables 480
Configure mapping tables 481
E.164 to E.164 Mapping Tables 485
Custom SIP URIs from called-/calling-e164 properties 488
Other mapping tables 488
Deleting mapping tables 489
Creating complex functions 490
Deleting complex functions 491
Digit collection & sending-complete behavior 492
Sending-Complete 492
Ingress interface 492
Call-Router 493
Egress Interface 495
Creating call services 497
Creating a hunt group service 497
Creating a distribution group service 506
Distribution-Group Min-Concurrent setting 508
Call-router 'limiter' service 508
Priority service 509
CS Bridge service—'VoIP Leased Line' 511
Configuring the service second-dialtone 513
Deleting call services 514
Activate the call router configuration 514
Test the call router configuration 515
Configure partial rerouting 521
Call reroute 522
Enable acceptation of rerouting requests on ISDN. 522
Enable emission of rerouting requests on ISDN. 522

Enable sending of "302 moved temporary" message on SIP. 522
Allow Push-Back 522
Enable push-back – aaa service 522
Enable push-back – bridge service 523
Enable push-back – distribution-group service 523
Enable push-back – hunt group service 523
Enable push-back – limiter service 523
Enable push-back – priority service 523

41 SIP call-router services 524
Introduction 525
SIP conference-service 525
SIP conference-service configuration task list 525
Entering conference-service configuration mode 525
Configuring the call routing destination 525
Configuring the conference server 526
SIP location-service 526
SIP location-service configuration task list 527
Entering SIP location-service configuration mode 527
Binding a location service 528
Configuring multi-contact behavior 528
Configuring the hunt timeout 528

42 Tone configuration 529
Introduction 530
Tone-set profiles 530
Tone configuration task list 531
Configuring call-progress-tone profiles 531
Configure tone-set profiles 532
Enable tone-set profile 533
Show call-progress-tone and tone-set profiles 534

43 FXS port configuration 537
Introduction 538
Shutdown and enable FXS ports 538
Bind FXS ports to higher layer applications 539
Configure country-specific FXS port parameters 539
Other FXS port parameters 540
Example 541

44 FXO port configuration 542
Introduction 543
Shutdown and enable FXO ports 543
Bind FXO ports to higher layer applications 543
Configure country specific FXO port parameters 544
Other FXO port parameters 544

45 H.323 gateway configuration 546
Introduction 547
Gateway configuration task list 548
Binding the gateway to an IP interface 548
Enable the gateway 548
Configure registration authentication service (RAS) (Optional) 549
Configure H.235 Security (optional) 550
H.235 configuration 551
Advanced configuration options (optional) 554
Enabling H.245 Tunneling 554
Enabling the fastconnect procedure 555
Enabling the early H.245 procedure 555
Changing the TCP port for inbound call-signaling connections 556
Configuring the traffic class for H.323 signaling 556
Setting the response timeout 556
Setting the connect timeout 557
Configuring the terminal type for registration with the gatekeeper 557
Troubleshooting 558
46 Context SIP gateway overview 559
Introduction 560
Context SIP Gateway configuration task list 561
Creating a context SIP gateway 561
Creating a transport interface 562
Configuring the IP binding 562
Configuring a priority 562
Configuring a spoofed contact address 563
Binding location services 563
Enabling/disabling the context SIP gateway 563
Troubleshooting 564
Show status information 564
Debug commands 564
Configuration Examples 565
Example 1 565
Example 2 565
Example 3 565
Applications 566
Outbound Authentication 566
Inbound Authentication 567
Outbound Registration 568
Inbound Registration 570
B2B User Agent with Registered Clients 571
47 VoIP profile configuration 573
Introduction 574

VoIP profile configuration task list 575
Creating a VoIP profile 575
Configure codecs 576
Configuring the transparent-clearmode codec 578
Configuring the Cisco versions of the G.726 codecs 578
Configuring DTMF relay 579
Configuring RTP payload types 579
Configuring RTP payload type for transparent-clearmode 580
Configuring RTP payload types for the g726-32k and g726-32k-cisco coders 580
Configuring RTP payload type for Cisco NSE 580
Configuring Cisco NSE for Fax 580
Configuring the dejitter buffer (advanced) 581
Enabling/disabling filters (advanced) 583
Configuring Fax transmission 584
T.38 CED retransmission 587
T.38 No-Signal Retransmission 588
Fax bypass method 588
Configuring fax failover 588
Configuring modem transmission 589
Modem bypass method 589
Configuring the traffic class for Voice and Fax data 590
Configuring IP-IP codec negotiation 590
Examples 591
Home office in an enterprise network 591
Home office with fax 593
Soft phone client gateway 594

48 PSTN profile configuration 597
Introduction 598
PSTN profile configuration task list 598
Creating a PSTN profile 598
Configuring the echo canceller 599
Configuring output gain 599

49 SIP profile configuration 601
Introduction 602
SIP profile configuration task list 602
Entering the configuration mode for a SIP profile 602
Mapping from a SIP disconnect cause 602
Mapping to a SIP cause 603
Mapping from a SIP redirection reason 603
Mapping to a SIP redirection code 603

50 Authentication Service 604
Introduction 605
Authentication Service configuration task list 605

Creating an Authentication Service 605
Configuring a Realm 606
Configuring the authentication protocol 606
Creating credentials 606
Configuration Examples 606

51 Location Service 607
Introduction 608
Location Service configuration task list 608
Creating a Location Service 608
Adding a domain 608
Creating an identity 609
Authentication outbound face 610
Authentication inbound face 611
Registration outbound face 613
Registration inbound face 615
Call outbound face 616
Call inbound face 617
Creating an identity group 618
Inheriting from an identity group to an identity 618
Configuring the Message Waiting Indication feature for SIP 619
Subscription 619
Notification 620
Configuration 620
Message Waiting Indication through Call-Control 622
Configuration Examples 623

52 VoIP debugging 624
Introduction 625
Debugging strategy 625
Filtering debug monitor output 626
Verifying IP connectivity 626
Debugging call signaling 627
Debugging ISDN signaling 627
Verify an incoming call 628
Verify an outgoing call 629
Verify ISDN layer 2 and 3 status 631
Debugging FXS Signaling 632
Verify an incoming call 632
Verify an outgoing call 633
Debugging H.323 Signaling 634
Verify an incoming call 634
Verify an outgoing call 636
Debugging SIP signaling 638
Verify an incoming call 638

Verify an outgoing call 638
Using SmartWare's internal call generator 639
Debugging voice data 640
Check system logs 642
How to submit trouble reports to Patton 642

A Terms and definitions 644
Introduction 645
SmartWare architecture terms and definitions 645

B Mode summary 650
Introduction 651

C Command summary 654
Introduction 655
New Configuration Commands 656
Other 656
Show help 656
Show command history 656
Restart system 656

D Internetworking terms & acronyms 657
Abbreviations 658

E Used IP ports & available voice codecs 662
Used IP ports 663
Available voice codecs 664

List of Figures

1 Basic system (abstract) model . . . 40
2 Typical carrier network application with a SmartNode. . . . 41
3 Typical enterprise network with SmartNode . . . 42
4 Typical LAN telephony system with a SmartNode gateway . . . 43
5 Configuration concept overview . . . 45
6 Setup for initial configuration via the console port . . . 55
7 Login display . . . 57
8 SmartNode memory regions logically defined in SmartWare . . . 68
9 Boot procedure . . . 74
10 Sample configuration file . . . 79
11 Local memory regions . . . 80
12 Remote memory regions for SmartWare . . . 82
13 System banner with message to operators . . . 94
14 Authentication procedure with a RADIUS server . . . 104
15 How to use AAA methods and AAA profiles . . . 104
16 IP context and related elements . . . 115
17 Dynamic NAPT . . . 134
18 Static NAPT . . . 134
19 Dynamic NAT . . . 135
20 Static NAT . . . 135
21 Binding of an Ethernet port to an IP interface . . . 144
22 Packet routing in SmartWare . . . 153
23 Example of Hierarchical Scheduling . . . 155
24 Elements of link scheduler configuration . . . 157
25 Scenario with Web server regarded as a single source host . . . 158
26 Structure of a Service-Policy Profile . . . 160
27 Using a Service Policy Profile on an IP Interface . . . 167
28 IP interface wan is bound to PVC 1 on port serial 0 0 . . . 183
29 Typical Integrated Service Access Scenario with dedicated PVCs . . . 187
30 IP Context with logical IP interfaces bound to Ethernet port, serial port PVC 1 and PVC 2 . . . 188
31 ISDN reference points . . . 213
32 ISDN signaling side . . . 214
33 Integration of ISDN access lines . . . 215
34 ISDN layering model . . . 216
35 PBX connected to ISDN port 1/0 . . . 224
36 Configuring the G.SHDSL card for PPPoE . . . 231
37 Internetwork with three routers and four networks . . . 240
38 Using traffic filters to prevent traffic from being routed to a network . . . 255
39 Deny a specific subnet on an interface . . . 266
40 AdventNet MibBrowser displaying some of the System Group objects . . . 272
41 AdventNet MibBrowser Settings Button on the Toolbar . . . 276
42 AdventNet TrapViewer displaying received traps . . . 277
43 AdventNet Trap Details window of TrapViewer . . . 278
44 DHCP-client and DHCP-server . . . 294
45 DNS relay diagram . . . 306
46 PPP configuration overview . . . 314
47 CS context configuration components . . . 340

48 Remote office in an Enterprise network . . . 342
49 Direct call routing from one SmartNode to another . . . 346
50 SmartNode in an Enterprise network . . . 351
51 CS Configuration . . . 352
52 CS interfaces on the CS context . . . 382
53 Incoming call passing an interface mapping table . . . 387
54 Call passing an input and an output mapping table . . . 388
55 ISDN interfaces on the CS context . . . 391
56 Example SIP network connecting two device to give a home office access to the CO PBX . . . 396
57 FXS interfaces on the CS context . . . 405
58 FXO interfaces on the CS context . . . 415
59 H.323 interfaces on the CS context . . . 432
60 SIP interfaces on the CS context . . . 442
61 Example SIP network connecting two devices to give a home office access to the CO PBX . . . 452
62 Direct call routing vs. advanced call routing . . . 459
63 Routing table outline . . . 464
64 Mapping table outline . . . 481
65 Mapping table examples . . . 484
66 Hunt group service . . . 498
67 Distribution group service . . . 506
68 Distribution group service examples . . . 507
69 'Limiter' service diagram . . . 509
70 Priority service diagram . . . 510
71 CS Bridge service—'VoIP Leased Line' diagram . . . 511
72 Bridge services diagram . . . 512
73 Call routing example network . . . 517
74 CS context and call router elements . . . 519
75 Registration and Lookup . . . 527
76 Assign tone-sets to a PSTN interfaces . . . 531
77 Gateway between IP and CS contexts . . . 547
78 Routing Architecture . . . 560
79 VoIP profile association . . . 574
80 DTMF Relay . . . 579
81 Jitter and dejitter buffer . . . 581
82 Adaptive versus static dejitter buffer . . . 582
83 Multiple tandem and sequential post filtering . . . 583
84 Fax relay and Fax bypass . . . 585
85 Home office in an enterprise network . . . 591
86 PSTN profile association . . . 598
87 Echo Cancellation . . . 599
88 Applying output gain . . . 599
89 Mode overview, 1 of 3 . . . 651
90 Mode Overview, 2 of 3 . . . 652
91 Mode Overview, 3 of 3 . . . 653
92 EBNF syntax . . . 655

List of Tables

1 General conventions 34
2 Mouse conventions 36
3 Command edit shortcuts 52
4 Command cross reference 156
5 TOS values and their meaning 163
6 Traffic control info (TCI) field 164
7 Values defining detail of the queuing statistics 169
8 PVC Commands 233
9 PVC channels in bridged Ethernet mode 233
10 PVC channels in PPPoE mode 233
11 Diagnostics commans 234
12 Details available in the Trap Details window 278
13 Time servers operated by NIST 290
14 FXS services with permanent patterns 410
15 FXS services with configurable patterns 410
16 ISDN number types 463
17 Routing table types 464
18 Wildcard symbols used as keys in E.164 tables (calling-e164, called-e164) 467
19 Wildcard symbols used as keys in E.164 tables (calling-e164, called-e164) 468
20 Mapping table types 482
21 Hunt group drop causes 500

About this guide

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

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 34)

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

How to read this guide

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

- Appendix A, "Terms and definitions" on page 644 contains the terms and their definitions that are used throughout this *SmartWare Software Configuration Guide*
- Chapter 1, "System overview" on page 38 provides an overview of the main elements of a SmartWare system.
- Chapter 9, "IP context overview" on page 114
- Chapter 31, "CS context overview" on page 339

Structure

This guide contains the following chapters and appendices:

- Chapter 1, "System overview" on page 38 provides an overview of the main elements of a SmartWare system.
- Chapter 2, "Configuration concepts" on page 44 introduces basic SmartWare configuration concepts.
- Chapter 3, "Command line interface (CLI)" on page 49 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 53 describes the procedures for entering SmartWare commands via the command line interface (CLI), to obtain help, to change operator mode and to terminate a session.
- Chapter 5, "System image handling" on page 65 describes how to load and maintain system images and driver software.
- Chapter 6, "Configuration file handling" on page 76 describes how to upload and download configuration files from and to a SmartNode.
- Chapter 7, "Basic system management" on page 90 describes parameters that report basic system information to the operator or administrator, and their configuration.
- Chapter 8, "RADIUS Client Configuration" on page 102 provides an overview of the authentication, authorization, and accounting (AAA) component in SmartWare and describes how to configure the RADIUS client, a subpart of the AAA component.
- Chapter 9, "IP context overview" on page 114 outlines SmartWare *Internet protocol* (IP) context, together with its related components.
- Chapter 10, "IP interface configuration" on page 120 provides a general overview of SmartNode interfaces and describes the tasks involved in their configuration.
- Chapter 11, "NAT/NAPT configuration" on page 132 provides a general overview of the network address port translation and describes the tasks involved in its configuration.
- Chapter 12, "Ethernet port configuration" on page 141 provides an overview of Ethernet ports and describes the tasks involved in their configuration through SmartWare.
- Chapter 13, "Link scheduler configuration" on page 151 describes how to use and configure SmartWare *quality of service* (QoS) features.
- Chapter 14, "Serial port configuration" on page 170 provides an overview of the serial port and describes the tasks involved in its configuration through SmartWare.
- Chapter 15, "Frame Relay configuration" on page 177 provides an overview of how to configure frame relay through SmartWare.
- Chapter 16, "PRI port configuration" on page 191 provides an overview of the T1/E1 ports, their characteristics and the tasks involved in the configuration.
- Chapter 17, "BRI port configuration" on page 205 provides an overview of the BRI (Basic Rate Interface) ports, their characteristics and the tasks involved in the configuration.
- Chapter 18, "ISDN Overview" on page 212 provides an overview of ISDN ports and describes the tasks involved in configuring ISDN ports in SmartWare.

- Chapter 19, "ISDN configuration" on page 217 describes the configuration of the Q.921 and Q.931 protocol and how to bind the ISDN protocol to an application.
- Chapter 20, "RBS configuration" on page 225 describes the configuration of the Robbed Bit Signaling (RBS) protocol and how to bind it to the Call Control application.
- Chapter 37, "RBS interface configuration" on page 426 provides an overview of RBS interfaces, and the tasks involved in their configuration.
- Chapter 21, "DSL Port Configuration" on page 230 provides an overview of the the DSL ports (ADSL and G.SHDSL), their characteristics and the tasks involved in the configuration.
- Chapter 22, "Basic IP routing configuration" on page 235 provides an overview of IP routing and describes the tasks involved in configuring static IP routing in SmartWare.
- Chapter 23, "RIP configuration" on page 242 provides an overview of the *routing information protocol* (RIP) and describes the tasks involved in configuring RIP features within SmartWare.
- Chapter 24, "Access control list configuration" on page 253 provides an overview of IP access control lists and describes the tasks involved in their configuration through SmartWare.
- Chapter 25, "SNMP configuration" on page 267 on page 238 provides overview information about the *simple network management protocol* (SNMP) and describes the tasks used to configure those of its features supported by SmartWare.
- Chapter 26, "SNTP client configuration" on page 282 describes how to configure a *simple network time protocol* (SNTP) client.
- Chapter 27, "DHCP configuration" on page 292 provides an overview of the *dynamic host configuration control protocol* (DHCP) and describes the tasks involved in its configuration.
- Chapter 28, "DNS configuration" on page 304 describes how to configure the *domain name system* (DNS) component.
- Chapter 29, "DynDNS configuration" on page 308 describes configuring the *dynamic DNS* (DynDNS) service.
- Chapter 30, "PPP configuration" on page 313 describes how to configure the *point-to-point protocol* over different link layers.
- Chapter 31, "CS context overview" on page 339 gives an overview of SmartWare *circuit-switching* (CS) context and its associated components and describes the tasks involved in its configuration.
- Chapter 32, "VPN configuration" on page 362 describes how to configure the VPN connections between two SmartNodes or between a SmartNode and a third-party device.
- Chapter 33, "CS interface configuration" on page 381 gives an overview of interfaces in the CS context and describes the tasks involved its configuration.
- Chapter 34, "ISDN interface configuration" on page 390 provides an overview of ISDN interfaces, and the tasks involved in their configuration.
- Chapter 35, "FXS interface configuration" on page 404 provides an overview of FXS interfaces, and the tasks involved their configuration.

- Chapter 36, "FXO interface configuration" on page 414 provides an overview of FXO interfaces and the tasks involved in configuring them.
- Chapter 38, "H.323 interface configuration" on page 431 provides an overview of H.323 interfaces used by H.323 gateways and describes the specific tasks involved in their configuration.
- Chapter 39, "SIP interface configuration" on page 441 provides an overview of SIP interfaces used by SIP gateways and describes the specific tasks involved in their configuration.
- Chapter 40, "Call router configuration" on page 456 provides an overview of call router tables, mapping tables and call services and describes the tasks involved in configuring the call router in SmartWare.
- Chapter 41, "SIP call-router services" on page 524 provides an overview of specific SIP call router services in SmartWare.
- Chapter 42, "Tone configuration" on page 529 gives an overview of SmartWare call-progress-tone profiles and tone-set profiles and describes the tasks involved in their configuration.
- Chapter 43, "FXS port configuration" on page 537 provides an overview of POTS signaling and SmartNode FXS ports and describes the tasks involved in configuring FXS ports in SmartWare.
- Chapter 44, "FXO port configuration" on page 542 provides an overview of POTS signaling and SmartNode FXO ports and describes the tasks involved in configuring FXO ports in SmartWare.
- Chapter 45, "H.323 gateway configuration" on page 546 provides an overview of the H.323 gateway and describes the tasks involved in its configuration.
- Chapter 46, "Context SIP gateway overview" on page 559 provides an overview of the SIP gateway and describes the tasks involved in its configuration.
- Chapter 47, "VoIP profile configuration" on page 573 gives an overview of SmartWare VoIP profiles, how they are used and describes the tasks involved in VoIP profile configuration.
- Chapter 48, "PSTN profile configuration" on page 597 gives an overview of SmartWare PSTN profiles, and describes how they are used and the tasks involved in PSTN profile configuration.
- Chapter 49, "SIP profile configuration" on page 601 gives an overview of mappings between SIP codes and Q.931 causes.
- Chapter 50, "Authentication Service" on page 604 explains how to create and manage authentication services in SmartWare.
- Chapter 51, "Location Service" on page 607 explains how to configure location services in SmartWare.
- Chapter 52, "VoIP debugging" on page 624 helps you to localize a system component that is responsible for faults during operation of a SmartNode device.
- Appendix A, "Terms and definitions" on page 644 contains the terms and their definitions that are used throughout this *SmartWare Software Configuration Guide*.
- Appendix B, "Mode summary" on page 650 illustrates the modes hierarchy.
- Appendix C, "Command summary" on page 654 is a command reference.
- Appendix D, "Internetworking terms & acronyms" on page 657 contains terms and definitions relating to internetworking.

- Appendix E, "Used IP ports & available voice codecs" on page 662 describes the used IP ports and available voice codecs in SmartWare.
- Appendix F, "Notes for upgrading from R3.10 to R3.20" on page 618 describes how to upgrade a SmartNode device from Release 3.10 to 3.20.

Precautions

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

Note A note presents additional information or interesting sidelights.

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

Typographical conventions used in this document

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

General conventions

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

Table 1. General conventions

Convention	Meaning
Garamond blue type	Indicates a cross-reference hyperlink that points to a figure, graphic, table, or section heading. Clicking on the hyperlink jumps you to the reference. When you have finished reviewing the reference, click on the **Go to Previous View** button in the Adobe® Acrobat® Reader toolbar to return to your starting point.
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 a SmartNode is substituted with ***node*** in ***boldface italic*** font.
node	The leading **node** on a command line represents the nodename of the SmartNode

Table 1. General conventions

Convention	Meaning
#	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 39
SmartWare embedded software 40
Applications 41
Carrier networks 41
Enterprise networks 42
LAN telephony 43

Introduction

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

A complete SmartNode system or network, as installed in any of the application scenarios introduced in section "Applications" on page 41, is typically composed of the following main elements plus a third-party network infrastructure:

- The first and most obvious element is the *SmartNode* devices (also referred to as *hardware platforms* or *network nodes*) that provide the physical connectivity, the CPU and DSP resources. All SmartNode models support packet-routed and circuit-switched traffic equally well.
- The second element comprises the embedded software—called *SmartWare*—running on the SmartNode hardware platforms.
- 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 depicts the basic system model of a Patton SmartNode. All SmartNode devices have the following main components:

- 64k circuit switching between on-board ISDN ports and between ISDN and PSTN interface cards. The circuit switching engine uses dedicated hardware resources and therefore can bypass the VoIP gateway and packet routing engine.
- A gateway (GW) that converts telephone circuits into Internet protocol (IP) packet streams and vice versa. H.323-compliant and SIP Voice over IP (VoIP) is supported.
- An IP router with on-board ports and optional data interface cards is QoS enabled, thereby allowing classification, shaping, and scheduling of multiple service classes.

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

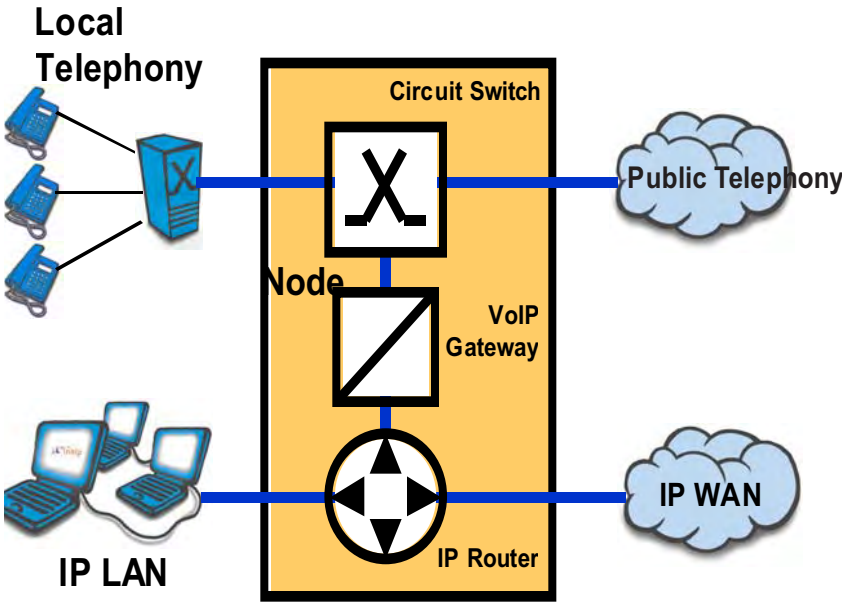


Figure 1. Basic system (abstract) model

SmartWare embedded software

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

A SmartWare build is a binary image file. It is usually divided into several checksum-protected files to improve download efficiency and security. The download to the SmartNode is handled in sequence by using a download *batchfile*. Refer to chapter 5, "System image handling" on page 65 for details on SmartWare image downloads.

Applications

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

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

Carrier networks

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

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

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

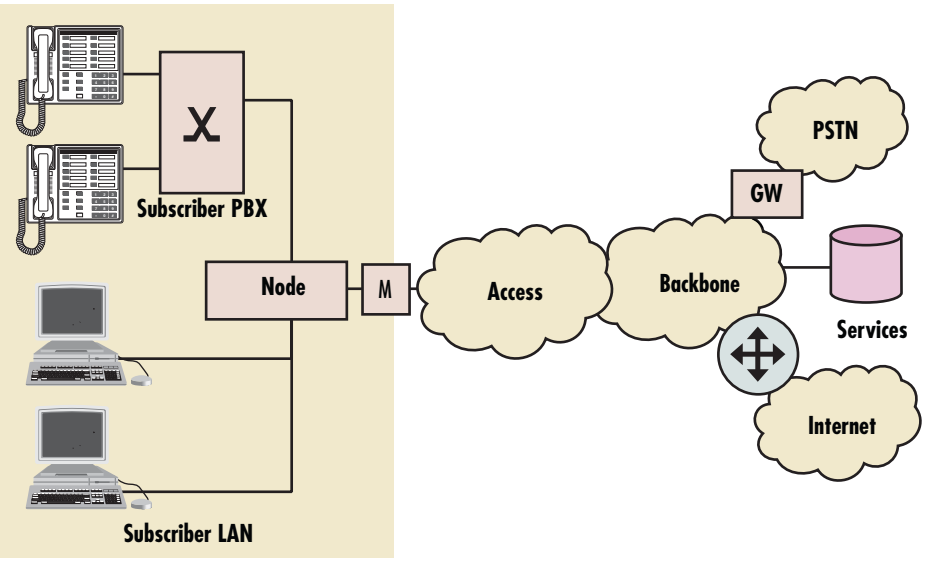


Figure 2. Typical carrier network application with a SmartNode.

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

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

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

Enterprise networks

In company-owned and operated wide area networks, SmartNodes can be used to converge voice and data communications on the same IP link.

In combination with centralized services such as groupware and unified messaging, the SmartNodes provide migration and investment protection for legacy telephony systems.

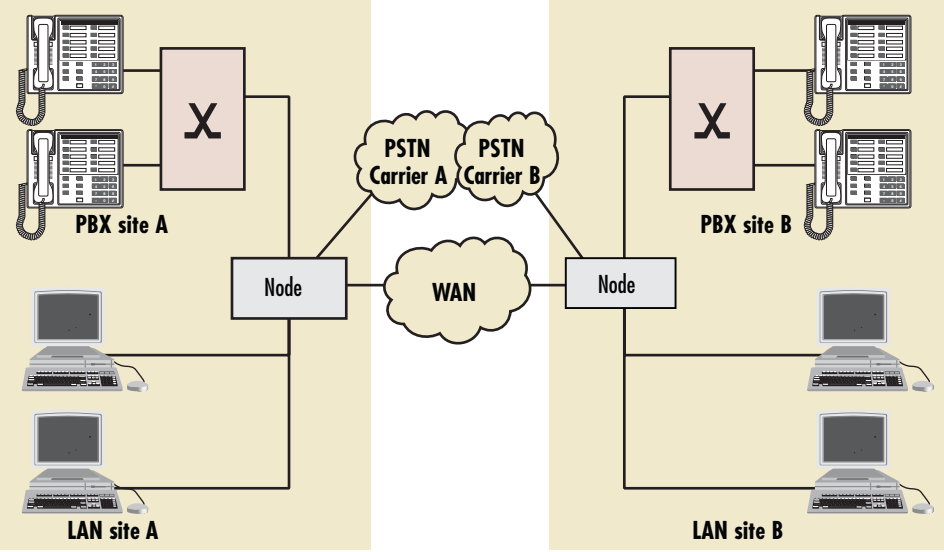


Figure 3. Typical enterprise network with SmartNode

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

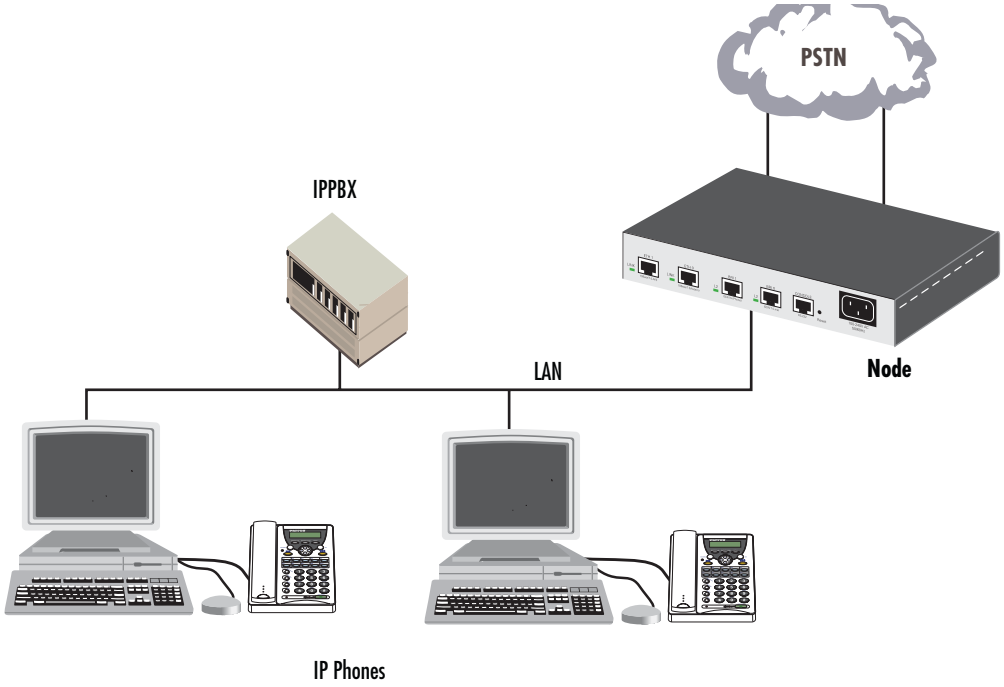


Figure 4. Typical LAN telephony system with a SmartNode gateway

LAN telephony

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

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

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

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

Chapter 2 Configuration concepts

Chapter contents

Introduction 45
Contexts and Gateways 46
Context 46
Gateway 46
Interfaces, Ports, and Bindings 47
Interfaces 47
Ports and circuits 47
Bindings 47
Profiles and Use commands 48
Profiles 48
Use Commands 48

Introduction

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

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

This chapter includes the following sections:

- Contexts and gateways (see page 46)
- Interfaces, ports, and bindings (see page 47)
- Profiles and Use commands (see page 48)

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

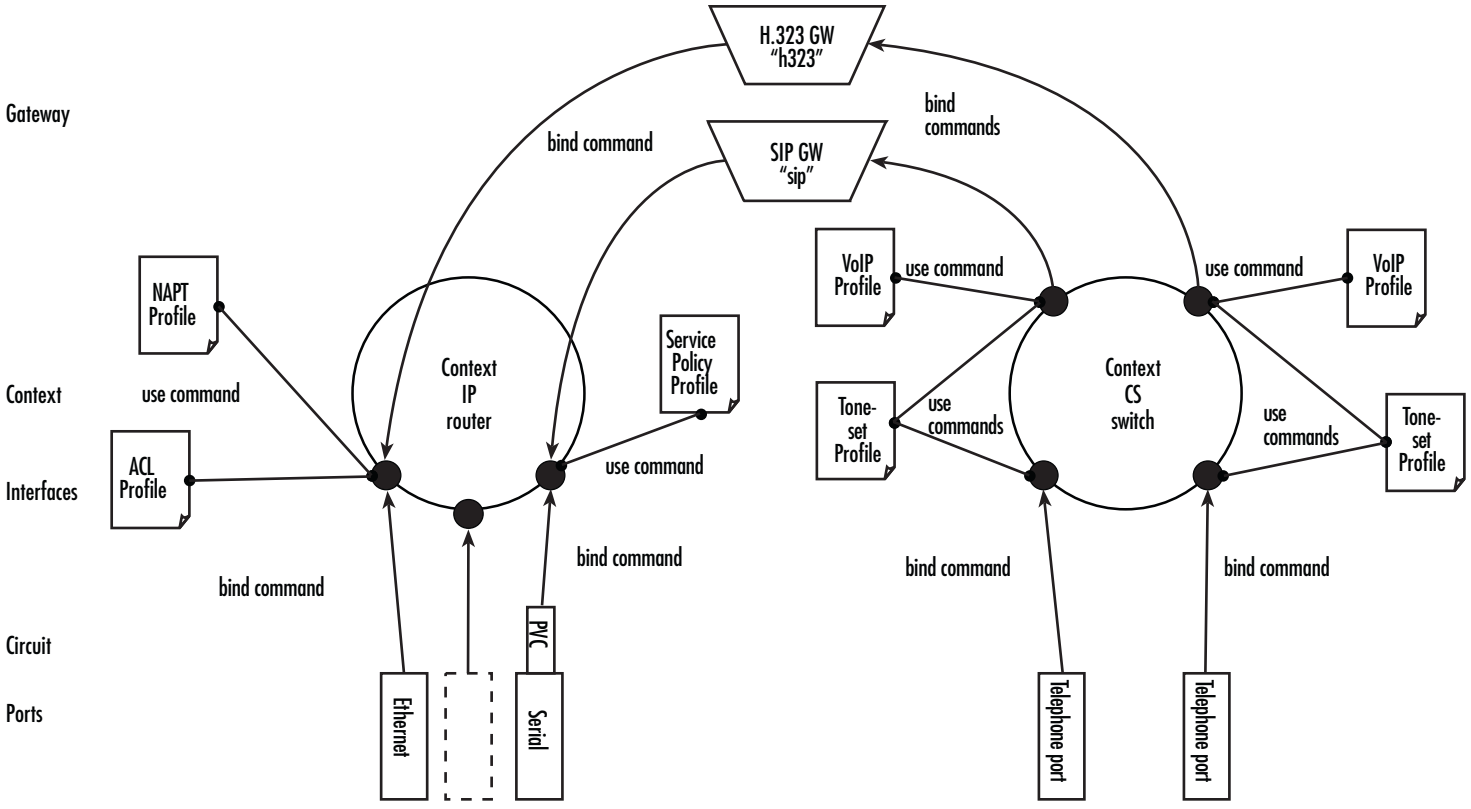


Figure 5. Configuration concept overview

Figure 5 shows the various elements of a complete SmartNode configuration. Each of these elements implements one of the configuration concepts described in this chapter. The figure also shows the relationships and associations between the different elements. The relations are specified through *bind* (arrow) and *use* (bullet-

lines) commands. For example, you need *bind* commands to bind a physical port to a logical interface, and *use* commands to assign profiles to contexts.

The sections that follow refer to figure 5 on page 45 and describe the concepts and elements in more detail.

Contexts and Gateways

Context

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

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

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

Each context contains a number of *interfaces*, which build the connections to other SmartWare elements and the outside world. Figure 5 on page 45 shows two contexts:

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

> **Note** SmartWare currently supports only one instance of the CS and IP context types.

Example

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

Gateway

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

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

Interfaces, Ports, and Bindings

Interfaces

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

In order for the higher-layer protocols to become active, you must associate an interface with a physical port or circuit. This association is referred to as a *binding* in SmartWare. Refer to the "Bindings" section for more information. In figure 5 on page 45, the IP context shows three interfaces and the CS context shows four interfaces. These interfaces are configured within their contexts. The bindings shown in the figure are not present when the interfaces are configured; they are configured later.

Ports and circuits

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

Examples of ports are: Ethernet, Serial, DSL, FXS or FXO. Ports are numbered according to the label (or abbreviation) printed on the hardware.

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

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

Example: frame-relay pvc 112.

Figure 5 on page 45 shows five ports. Three ports are bound directly to an IP interface. One port has a single circuit configured, which is bound to the IP context. Two ISDN ports are bound to CS interfaces.

Bindings

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

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

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

Bindings from ports to interfaces shown in figure 5 on page 45.

Profiles and Use commands

Profiles

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

Profiles used in the IP and CS contexts are shown in figure 5 on page 45.

Use Commands

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

Chapter 3 Command line interface (CLI)

Chapter contents

Introduction 50
Command modes 50
CLI prompt 50
Navigating the CLI 51
Initial mode 51
System changes 51
Configuration 51
Changing Modes 51
Command editing 51
Command help 51
The No form 51
Command completion 51
Command history 52
Command Editing Shortcuts 52

Introduction

The primary user interface to SmartWare is the command line interface (CLI). You can access the CLI via the SmartNode console port or through a Telnet session. The CLI lets you configure the complete SmartWare functionality. 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 51)

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 650 contains a detailed overview of all command modes, and appendix C, "Command summary" on page 654 describes the commands that are valid in each mode.

CLI prompt

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

```
nodename>
```

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

```
nodename#
```

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

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

Where:

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

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

```
node(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 650.

Navigating the CLI

Initial mode

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

System changes

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

Configuration

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

Changing Modes

The **exit** command moves the user up one level in the mode hierarchy (the same command works in any of configuration modes). 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 DHCP server trace tool, enter the command **debug dhcp-server**. To subsequently disable the DHCP server trace, enter the command **no debug dhcop-server**.

Command completion

You can use the **<tab>** key in any mode to carry out command completion. Partially typing a command name and pressing the **<tab>** key causes the command to be displayed in full up to the point where a further choice has to be made. For example, rather than typing **configure**, typing **conf** and pressing the **<tab>** key causes the

CLI to complete the command at the prompt. If the number of characters is not sufficient to uniquely identify the command, the CLI will provide a list with all commands starting with the typed characters. For example, if you enter the string *co* in the configure mode and press **<tab>**, the selections **configure**, **copy**, and **context** are displayed.

Command history

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

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

Command Editing Shortcuts

SmartWare CLI provides a number of command shortcuts that facilitate editing of the command line. Command editing shortcuts are summarized in table 3 on page 52. 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 3. 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.
<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 54
Accessing the SmartWare CLI task list 54
Accessing via the console port 55
Console port procedure 55
Telnet Procedure 56
Using an alternate TCP listening port for the Telnet server 56
Disabling the Telnet server 56
Logging on 56
Selecting a secure password 57
Password encryption 58
Factory preset administrator account 58
Creating an operator account 58
Creating an administrator account 59
Opening a secure configuration session over SSH 59
Displaying the CLI version 60
Displaying account information 60
Switching to another account 61
Checking identity and connected users 61
Command index numbers 62
Ending a Telnet or console port session 64
Showing command default values 64

Introduction

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

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

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

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

Remember that the CLI supports a command history and command completion. By scrolling with the **up** and **down** arrow keys, you can find many of your previously entered commands. Another timesaving tool is command completion. If you type part of a command and then press the **<tab>** key, the SmartWare shell will present you with either the remaining portion of the command or a list of possible commands. These features are described in chapter 3, "Command line interface (CLI)" on page 49. The telnet server can be disabled if desired.

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

Accessing the SmartWare CLI task list

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

- Accessing via the console port (see page 55)
- Accessing via a Telnet session (see page 55)
- Using an alternate TCP listening port for the Telnet server (see page 56)
- Disabling the Telnet server (see page 56)
- Logging on (see page 56)
- Selecting a secure password (see page 57)
- Configuring operators and administrators (see page 58)
- Displaying the CLI version (see page 60)
- Displaying account information (see page 60)
- Switching to another log-in account (see page 61)
- Checking identity and connected users (see page 61)

- Ending a Telnet or console port session (see page 64)

Accessing via the console port

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

Figure 6. Setup for initial configuration via the console port

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

Console port procedure

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

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

Accessing via a Telnet session

This is the most commonly used and recommended method for connecting to a SmartNode. It is way faster than console access.The Telnet host accesses the SmartNode via its network interface.

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

Telnet Procedure

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

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

Using an alternate TCP listening port for the Telnet 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 on

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

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

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

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 unit.

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 SmartNode device!

Password encryption

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

- Plain text
- Encrypted text (for example, the password mypassword always appears in encrypted form as *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

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

Configuring operators and administrators

Creating an operator account

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

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

Mode: Operator execution

Step	Command	Purpose
1	***node*>enable**	Enters administration execution mode
2	***node*#configure**	Enters configuration mode
3	***node(cfg)#* operator** *name* **password** *password*	Creates a new operator account *name* and password *password*
4	**copy running-config startup-config**	Saves the change made to the running configuration of the SmartNode, so that it will be used following a reload

Example: Create an operator account

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

```
node>enable
node#configure
node(cfg)#operator support password s4DF&qw
```

```
node(cfg)#copy running-config startup-config
```

Creating an administrator account

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

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

Mode: Operator execution

Step	Command	Purpose
1	***node*>enable**	Enters administration execution mode
2	***node*#configure**	Enters configuration mode
3	***node(cfg)*# administrator** *name* **password** *password*	Creates a new administrator account *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*.

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

Opening a secure configuration session over SSH

A partial implementation of secure shell according RFC 4251, RFC 4252, RFC 4253 and RFC 4254 is provided. It is possible to open a secure configuration session over SSH to a SmartNode.

> **Note** The Web-GUI and the copy tftp function are still unsecure!

The SSH Transport Layer supports the following Algorithms: “ssh-rsa” public key for signing, “diffie-hellmann-group1-sha1” and “diffie-hellmann-group14-sha1” for key exchange, “3des-cbc”, “aes256-cbc”, “aes192-cbc” and “aes128-cbc” for encryption, “hmac-sha1” and “hmac-md5” for data integrity. For user authentication, only the method “password” is supported. On the Connection Layer, only the request for an interactive command shell is supported.

After the first startup of SmartWare, the RSA server host key is going to be calculated. This calculation is done in the background and with low priority, so that the SmartNode can operate normally. Until the RSA server host key is calculated, which takes several minutes, it is not possible to open SSH sessions. The RSA server host key is calculated only once and always remains the same.

Mode: Configure

Step	Command	Purpose
1	[name]**(cfg)#terminal ssh use authentication <AAA profile name>**	Set the AAA profile which is going to be used for user authentication. The AAA profile "default" is uses as when not specified otherwise.

Mode: Enable

Step	Command	Purpose
1	[name]**#show ssh**	Displays status information of the SSH server.

Mode: Enable

Step	Command	Purpose
1	[name]**#debug ssh**	Prints debug information of the SSH server.

Displaying the CLI version

This procedure displays the version of the currently running 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 CLI on your device, if you start from the operator execution mode.

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

Displaying account information

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

The following procedure describes how to display account information:

Mode: Administrator execution

Step	Command	Purpose
1	***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.

```
node#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>
node>who
You are operator support
node>su super
Enter password: <password>
node>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:

```
node#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:

```
node>who
You are operator support
```

Command index numbers

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

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

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

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

Listing before changing the G.723 codec index number:

```
profile voip default
  codec 1 g711ulaw64k rx-length 20 tx-length 20
  codec 2 g711alaw64k rx-length 20 tx-length 20
  codec 3 g723-6k3 rx-length 30 tx-length 30
  dejitter-max-delay 200
...
```

Listing after changing index number:

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

Note Succeeding indexes are automatically renumbered.

Example 2: Moving the G.723 codec back position 3
This command moves the G.723 codec from the top to third place. As a result, the other two codecs move up in the list as their indexes are automatically renumbered to accommodate the new third-place codec.

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

Example 3: Inserting a codec at a specific position in the list.
This command assigns the G.729 codec the index number 1 so the codec appears at the top of the list.

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

Ending a Telnet 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.

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

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

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

Showing command default values

If a command is set to its default value, it is not displayed in the running-config in order to make it more readable. There are a few exceptions to this rule. The command cli config defaults makes commands also appearin the running-config that are set to default values. no li config defaults turns it off.

Chapter 5 System image handling

Chapter contents

Introduction 66
Memory regions in SmartWare 67
System image handling task list 68
- Displaying system image information 69
- Copying system images from a network server to Flash memory 69
- Upgrading the software directly 71

Auto provisioning of firmware and configuration 72
Boot procedure 74
Factory configuration 75
- Default Startup Configuration 75
- IP Addresses in the Factory Configuration 75

Introduction

This chapter describes how to load, maintain, and update the various software images in the SmartNode. The SmartWare 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 SmartNode. Driver images are used to operate the various optional PMC interface cards.

This chapter includes the following sections:

- Memory regions in Smartware
- System image handling task list (see page 68)
- Boot procedure and bootloader (see page 74)

> **Note** Section "System image handling task list" on page 68 describes the standard way to upgrade the SmartWare. If you encounter problems that won't let you upgrade using the standard method, refer to section "Factory configuration" on page 75.

> **Note** Refer to appendix F, "Notes for upgrading from R3.10 to R3.20" on page 618 for information on converting from SmartWare release R3.10 to R3.20

- Factory configuration (see page 75)

Patton SmartNode 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 SmartNode at the factory. This configuration file sets the initial basic operating parameters of the SmartNode, such as enabling the Ethernet ports, setting the default IP addresses and the DHCP server.

Other configuration files may be stored in the SmartNode 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 SmartNode IP interfaces. Tftp cannot be used from the console interface.

The following sections focus on SmartWare memory regions, as well as the software components you can copy into the memory or move between a TFTP server and the memory of the SmartNode. As SmartWare uses a specific vocabulary in naming those software components, refer to appendix A, "Terms and definitions" on 644 to ensure that you understand the concepts.

Memory regions in SmartWare

The SmartNode's memory contains several logical regions and several physical regions as shown in figure 8 on page 68, 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 SmartNode. The command syntax in SmartWare 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 SmartNode. 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 SmartNode the *Volatile* physical region contains one logical region, *system:*, which is random access memory (RAM). When no power is applied to the SmartNode, 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 SmartNode 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 SmartNode. It is used—if no user-specific configuration is available—to start-up SmartWare with a minimal functionality. This configuration is named *factory-config* in SmartWare terminology.

On powering up a SmartNode (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 SmartNode 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 SmartNode, 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 SmartNode, 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 SmartNode.

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 SmartNode is copied into the volatile memory of the SmartNode prior to normal operation. This procedure takes place after the system bootstrap, where the application image (i.e. SmartWare) is started and a configuration must be available. Shortly before SmartWare 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*.

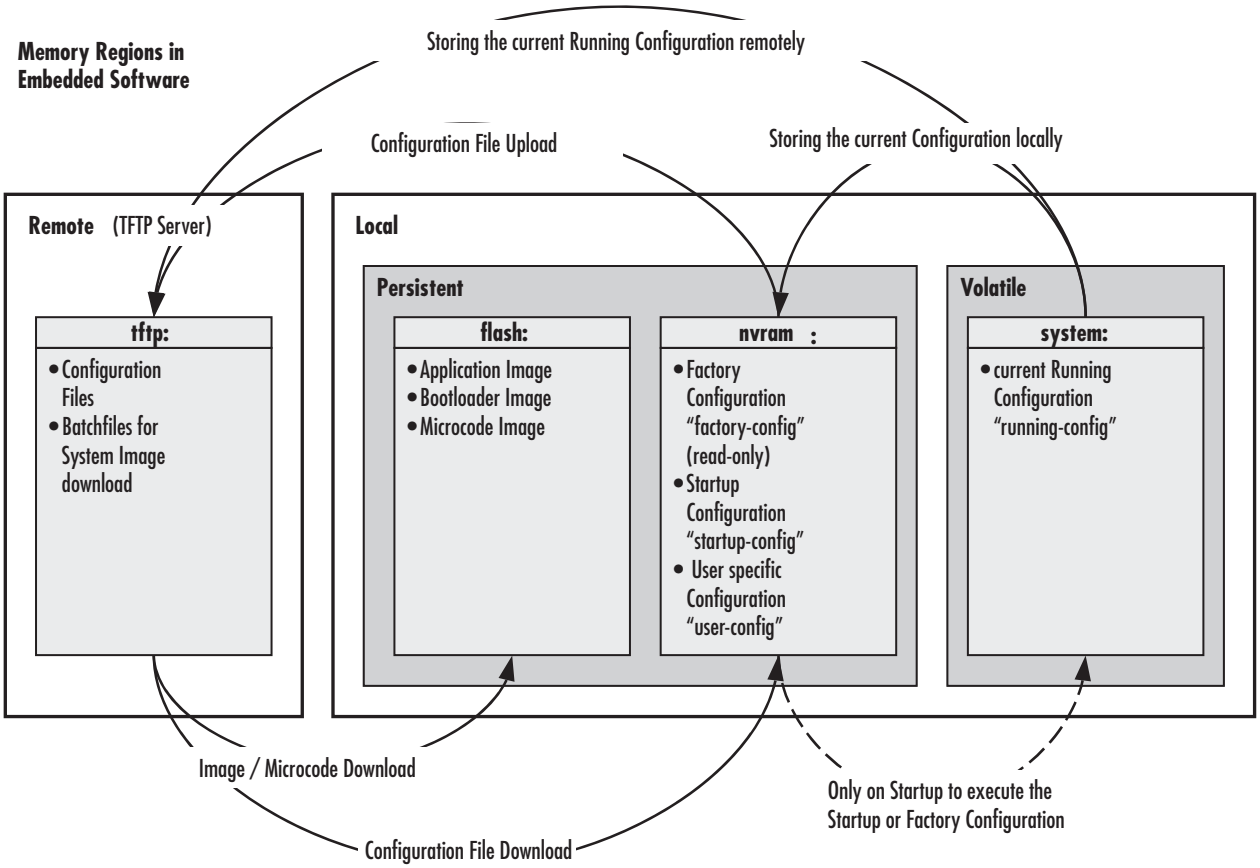


Figure 8. SmartNode memory regions logically defined in SmartWare

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 69)
- Copying the driver software from a network server to the Flash memory (see page 71)

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 a SmartNode 2000 series device with an optional IC-4BRV interface card mounted in slot 2.

```
node#show version

Productname      : SN4638/5BIS/UI
Software Version : R3.T 2006-12-04 H323 SIP BRI
Supplier         :
Provider         :
Subscriber       :

Information for Slot 0:
SN4638/5BIS/UI (Admin State: Application Started, Real State:
Application Started)
Hardware Version : 1, 3
Serial number    : 00A0BA0209B1
PLD Version      : 0x46010102
Software Version : R3.T 2006-12-04 H323 SIP BRI
```

Copying system images from a network server to Flash memory

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

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

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

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

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

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

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

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

SN(cfg)#
```

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

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

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

Upgrading the software directly

It is possible to upgrade the software directly by passing the name of the delivered zip-file to the CLI command "copy". The SmartWare downloads the whole ZIP file. During this time the download progress is displayed in bytes. After downloading, the ZIP file containing batch file "bw" or "b" will be extracted and executed. This leads to writing the SmartWare image, which is also part of the ZIP file, to the flash. The web pages are updated too. After writing the image to the flash, the Smartware needs to be reloaded with the command **reload**.

Mode: enable

Step	Command	Purpose
1	***node*(cfg)# copy tftp://<server-ip-address>/<path>/<smartwaredeliveryfile>.zip :flash**	Downloads the specified delivery file from the TFTP server and starts the driver software image upgrade process.

Example: An example of such a Smartware upgrade session, where the new software is in the file SN1000_SIP_R3.T_2006-08-10.zip which is stored on a tftp-server with the ip address 192.186.22.44:

```
node#copy tftp://192.186.22.44/SN1000_SIP_R3.T_2006-08-10.zip flash:
Download...   3124510 Bytes
Downloading image...completed (2715796 bytes)
Erasing flash...completed.
Writing to flash...completed
Processing files...completed
node#reload
```

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.

Step	Command	Purpose
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. $(system.mac) is a placeholder for the unit's MAC address of ETH 0/0. Using host names instead of IP addresses works only if DNS resolver is enabled and configured.
3	[name] **(pf-prov)**[CONFIG]**#location 2 tftp://172.16.1.33/configs/$(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 a SmartNode, the bootstrap application checks for an application image in the persistent memory of the logical region *nvram:*. The application image is then executed, i.e. the SmartWare is started module by module. 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 SmartWare now uses the *running-config* to set up the operating configuration of the SmartNode. Figure 9 illustrates the boot procedure.

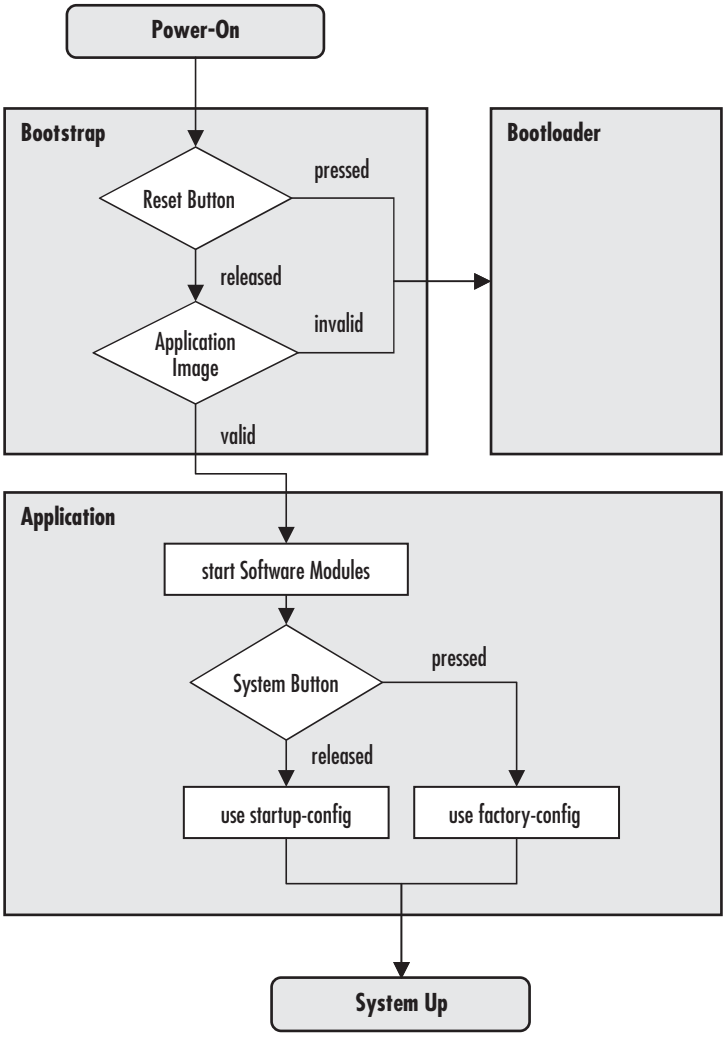


Figure 9. Boot procedure

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

Factory configuration

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

Once the user-specific configuration is created and stored as startup configuration, the factory configuration is no longer used but it remains in the persistent memory. It is possible to switch back to the factory configuration at any time during the operation of a SmartNode.

Default Startup Configuration

The SmartNodes delivered from the factory contain both a factory configuration and a default startup configuration. While the factory configuration contains only basic IP connectivity settings, the default startup configuration includes settings for most SmartWare functions. Note that if you press and hold the system button (Reset) for 5 seconds the factory configuration is copied onto the startup configuration (overwrite). The default startup config is then lost.

IP Addresses in the Factory Configuration

The factory configuration contains the following IP interfaces and address configurations bound by the Ethernet ports 0/0 and 0/1:

```
interface eth0
  ipaddress dhcp
  mtu 1500
interface eth1
  ipaddress 192.168.1.1 255.255.255.0
  mtu 1500
```

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

Chapter 6 Configuration file handling

Chapter contents

Introduction 77
Understanding configuration files 77
Factory configuration 79
Configuration file handling task list 79
Copying configurations within the local memory 80
Replacing the startup configuration with a configuration from Flash memory 81
Copying configurations to and from a remote storage location 82
Replacing the startup configuration with a configuration downloaded from TFTP server 83
Displaying configuration file information 83
Modifying the running configuration at the CLI 84
Modifying the running configuration offline 85
Deleting a specified configuration 86
Encrypted file download 87
Encrypted Configuration Download 87
Use Cases 88

Introduction

This chapter describes how to upload and download configuration files from and to SmartWare. A configuration file is a batch file of SmartWare commands used in the software modules that perform specific functions of the SmartNode. This chapter also describes some aspects of configuration file management. Refer to chapter 5, "System image handling" on page 65 for more information.

This chapter includes the following sections:

- Factory configuration (see page 79)
- Configuration file handling task list (see page 79)

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

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

Changing the current running configuration is possible as follows:

- You may change the running configuration interactively. Interactive configuring requires that you access the CLI by using the **enable** command to enter administrator execution mode. You must then switch to the configuration mode with the command **configure**. Once in configuration mode, enter the configuration commands that are necessary to configure your SmartNode.
- You can also create a new configuration file or modify an existing one offline. You can copy configuration files from the flash memory to a remote server. Transferring configuration files between the flash memory and a remote system requires the Trivial File Transfer Protocol (TFTP). The TFTP server must be reachable through one of the SmartNode network interfaces.

See chapter 4, "Accessing the CLI" on page 53 for information concerning access to the CLI.

The following sections focus on SmartWare memory regions and software components that can be copied within the memory or uploaded/downloaded between a TFTP server and the memory of the SmartNode. Since SmartWare uses a specific vocabulary in naming those software components, refer to appendix A, "Terms and definitions" on page 644 to ensure that you understand the concepts. Refer to chapter 5, "System image handling" on page 65 for a brief description of how SmartWare uses system memory.

Understanding configuration files

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

Figure 10, shows the characteristics of a configuration file. It is stored on a TFTP server in the file *myconfig.cfg* for later download. 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.

```
#-----------------------------------------------------------------#
# My Configuration File
#-----------------------------------------------------------------#


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

# system definitions
system
  clock-source 1 2
  hostname node

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

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

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

# CS context configuration
context cs switch
  no shutdown

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

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

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

```
  bind gateway h323

# H.323 gateway primarily used
  gateway h323
  faststart
  no ras
  gatekeeper-discovery auto
  bind interface lan router
  no shutdown

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

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

Figure 10. Sample configuration file

Each configuration file stored in the flash memory needs a unique name. The user has to assign a file name to any user-specific configuration. SmartWare predefines some names for configuration files. These are the factory configuration (*factory-config*), startup configuration (*startup-config*), and running configuration (*running-config*) file names. Refer to appendix A, "Terms and definitions" on page 644 to learn more about configuration file types.

Factory configuration

SmartNodes are delivered with a *factory configuration* in the logical region *nvram:*. This factory configuration initially parameterizes the most useful network and component settings of SmartWare.

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 a SmartNode configuration. The getting started guide included with your SmartNode device describes the restoration procedure for restoring the default settings.

Configuration file handling task list

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

The tasks in this chapter assume that you have at least a minimal configuration running on your system. You can create a basic configuration file by using the **configure** command; see section "Modifying the running configuration at the CLI" on page 84 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 80)
- Replacing the startup configuration with a configuration from the Flash memory (see page 81)
- Copying configurations to and from a remote storing location (see page 82)
- Replacing the startup configuration with a configuration downloaded from the TFTP server (see page 83)
- Displaying configuration file information (see page 83)
- Modifying the running configuration at the CLI (see page 84)
- Modifying the running configuration offline (see page 85)
- Deleting a specified configuration (see page 86)
- Downloading encrypted files (see page 87)

Copying configurations within the local memory

Configuration files may be copied into the local memory in order to switch between different configurations. Remember the different local memory regions in SmartWare as shown in figure 11.

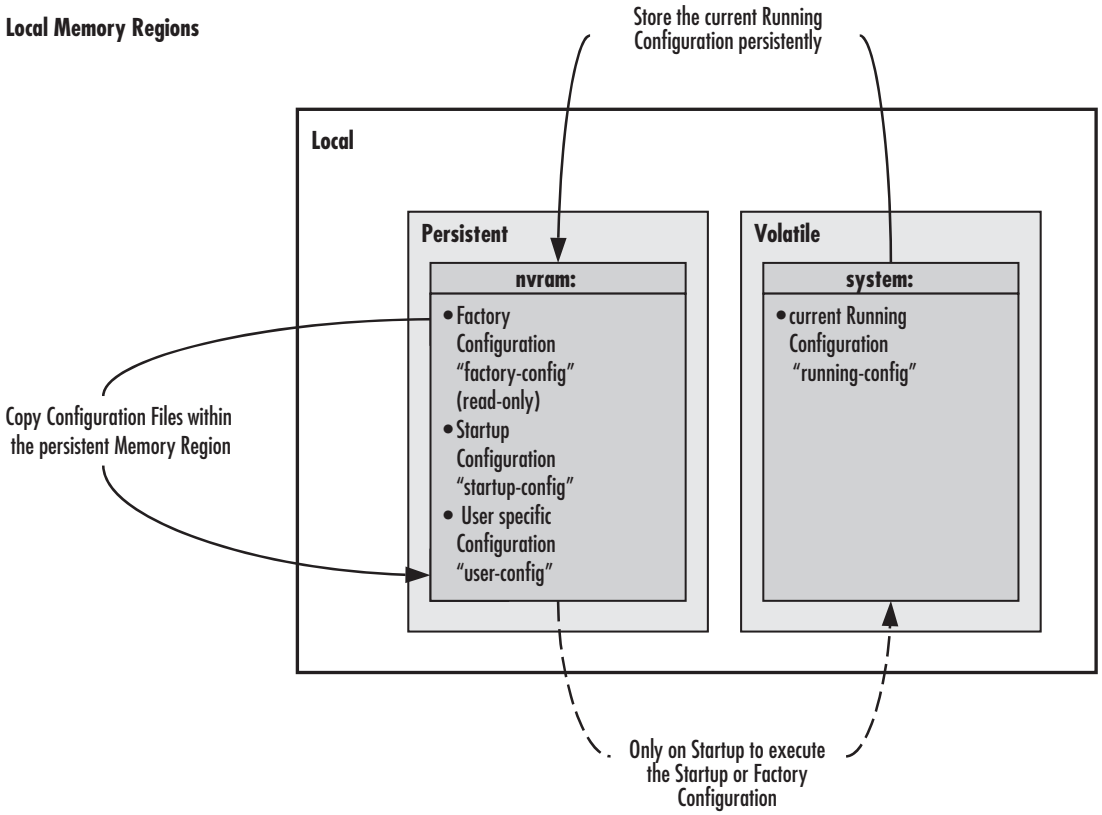


Figure 11. Local memory regions

In most cases, the interactively modified running configuration known as the *running-config*, which is located in the volatile memory region *system:*, is copied into the persistent memory region *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 11 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:.

```
node#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:***backup* **startup-config**	Replaces the existing persistent startup configuration with the startup configuration *backup* already present in flash memory.

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

Copying configurations to and from a remote storage location

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

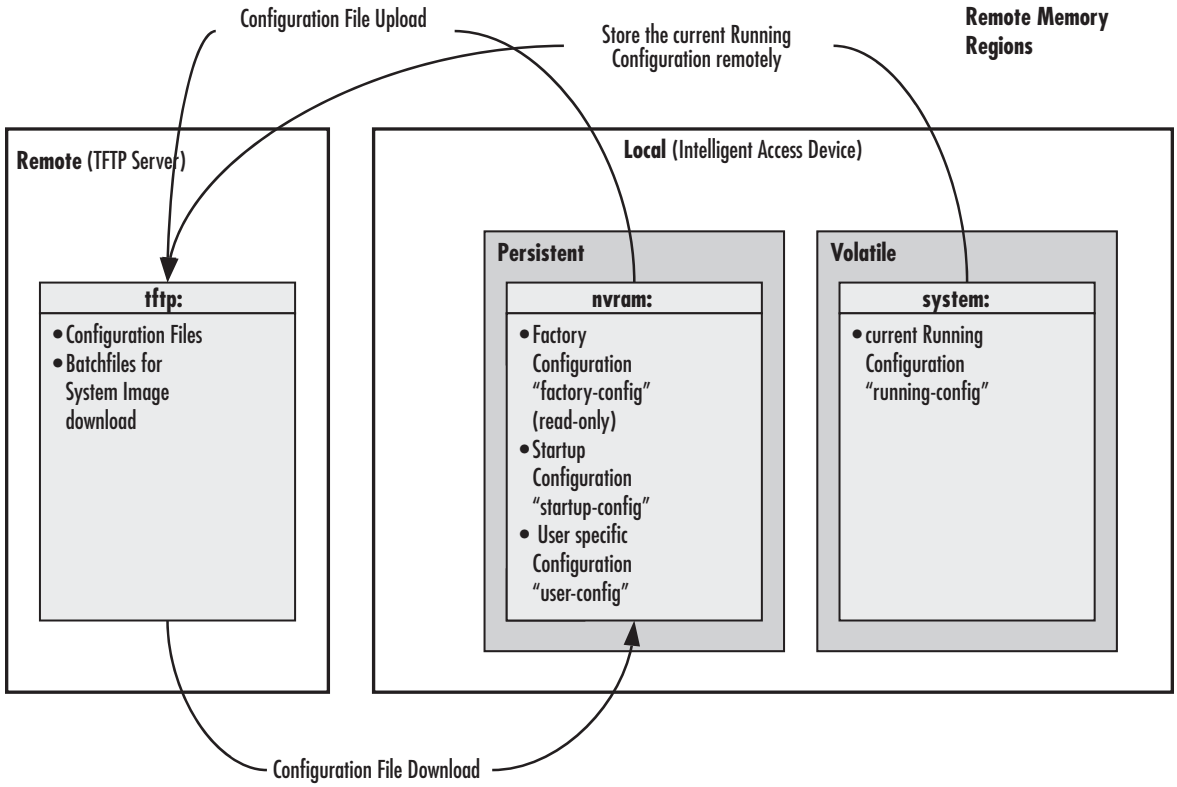


Figure 12. Remote memory regions for SmartWare

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

Replacing the startup configuration with a configuration downloaded from TFTP server

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

Mode: Administrator execution

Step	Command	Purpose
1	**node(cfg)# copy tftp://***ip-address[:port]/new-startup* **nvram:startup-config**	Downloads the configuration file *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 a SmartNode by overwriting it with the configuration contained in the file *new-startup* located on the TFTP server at IP address 172.16.36.80.

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

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

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

```
node#show nvram:startup-config
```

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

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

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

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

The execution command **reload** accepts 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.

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

Modifying the running configuration offline

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

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

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 SmartNode to the file *current-config* on a TFTP server at IP address 172.16.36.80. The uploaded configuration file is written into the root directory specified by the TFTP server settings, and overwrites any existing file with the same name. Read your TFTP server manual to get a thorough understanding of its behavior. After this, the configuration file is available for offline editing on the TFTP server. Once the configuration file *current-config* has been modified, it is downloaded from the TFTP server, at IP address 172.16.36.80, into the persistent memory region *nvram:* using the name *startup-config*. It will become active after a reload.

```
node#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.

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

Deleting a specified configuration

This procedure describes how to delete configuration files from the SmartNode flash memory region *nvram:*.

Mode: Administrator execution

Step	Command	Purpose
1	***node*#show nvram:**	Lists the loaded configurations
2	***node*#erase name**	Deletes the configuration *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.

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

2. Delete the configuration named *minimal* explicitly.

```
node#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.

```
node#show nvram:
Persistent configurations:
backup
```

```
startup-config
factory-config
```

Encrypted file download

This section explains how configuration files can be transported encrypted over IP.

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, ect.—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 SmartNode. The configurations can be stored in plain form on the SmartNode. 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 SmartNode the encryption is detected and the configuration file is automatically decrypted before stored to flash.

A custom encryption key can be:

- Downloaded to the SmartNode
- Specified with the PC encryption tool

The encryption key may include the MAC address and/or serial number of the SmartNode 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 SmartNode. 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 SmartNode.

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 SmartNode using the **copy** command of the SmartNode.

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 a SmartNode. Using this mechanism you can specify device-specific encryption keys. We currently support the following variables:

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

```
[...]
Serial number : 100000020002
[...]
```

The encryption key on this SmartNode will be interpreted as:

```
123100000020002abc000cf187d909XYZ
```

Then you have to download the created key file to the SmartNode. 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.

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 in the startup-config.

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 SmartNode 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 SmartNode 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 7 Basic system management

Chapter contents

Introduction 91
Basic system management configuration task list 91
Managing feature license keys 92
Setting system information 93
Setting the system banner 94
Setting time and date 95
Display clock information 95
Display time since last restart 96
Configuring the Web server 96
Determining and defining the active CLI version 96
Restarting the system 97
Displaying the system logs 97
Displaying reports 98
Controlling command execution 98
Timed execution of CLI command 100
Displaying the checksum of a configuration 100
Configuration of terminal sessions 100

Introduction

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

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

Additionally, the following tasks are described in this chapter:

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

Basic system management configuration task list

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

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

- Managing feature license keys (see page 92)
- Setting system information (see page 93)
- Setting the system banner (see page 94)
- Setting time and date (see page 95)
- Displaying clock information (see page 95)
- Displaying time since last restart (see page 96)
- Configuring and starting the web server (see page 96)
- Determining and defining the active CLI version (see page 96)
- Restarting the system (see page 97)
- Displaying the system event log (see page 97)
- Controlling command execution (see page 98)
- Setting timed execution of CLI commands (see page 100)
- Displaying the checksum of a configuration (see page 100)
- Configuration of terminal sessions (see page 100)
- Identifying a unit by flashing all LED's (see page 100)

Managing feature license keys

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

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

```
node(cfg)#copy tftp://172.16.4.3/keystore/myLicense.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.

```
node(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.

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

Setting system information

The system information includes the following parameters:

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

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

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

The system name, also called the hostname, is used to uniquely identify the SmartNode in your network. The selected name should follow the rules for ARPANET hostnames. Names must start with a letter, end with a letter or digit, and have as interior characters only letters, digits, and hyphens. Names must be 63 characters or fewer. For more information, refer to RFC 1035. This entry corresponds to the MIB II system sysName object. After setting the hostname of the SmartNode the CLI prompt will be replaced with the chosen name.

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

The system provider information is used to identify the provider contact for this SmartNode device, together with information on how to contact this provider. The provider is a company making services available to subscribers. The provider information may be any alphanumeric string, including spaces, that is no longer than one line. This entry corresponds to the Patton Electronics enterprise-specific MIB provider object.

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

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

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#system contact** *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.

```
node(cfg)#system contact "Bill Anybody, Phone 818 700 1504"
node(cfg)#system hostname node
node(cfg)#system location “Wiring Closet, 3rd Floor”
node(cfg)#system provider “Best Internet Services, contact@bis.com, Phone 818 700
2340”
node(cfg)# system subscriber “Mechanical Tools Inc., jsmith@mechtool.com, Phone 818
700 1402”
node(cfg)# system supplier “WhiteBox Networks Inc., contact@whitebox.com, Phone 818
700 1212”
```

Setting the system banner

The system banner is displayed on all systems that connect to your SmartNode via Telnet or a serial connection (see figure 13). 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 13. 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.

```
node(cfg)#banner \n#\n# The password of all operators has changed\n# please contact
the administrator\n#"
```

Setting time and date

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

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#clock set** *yyyy-mm-ddThh:mm:ss*	Sets the system clock to *yyyy-mm-ddThh:mm:ss*

Note The integrated SNTP client allows synchronization of time-of-day and date to a reference time server. Refer to chapter 26, "SNTP client configuration" on page 282 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.

```
node(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.

```
node>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.

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

Configuring the Web server

The embedded web server has two parameters that are configurable.

> **Note** Changing the language parameter does not affect the language of the web configuration pages.

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.

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

Determining and defining the active CLI version

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

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#show version cli**	Displays the currently running CLI version
2	***node*(cfg)#cli version** *version.revision*	Selects the active CLI version in the form version.revision

Example: Defining the desired CLI version

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

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

Restarting the system

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

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

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.

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

Displaying the system logs

The system logs contain warnings and information from the system components of SmartWare. In case of problems it is often useful to check the event or the supervisor logs for information about malfunctioning system components. The event log stores general events such as flash full, DSP failed etc., comparable with the event log on Windows NT. The supervisor log stores information from the system supervisor such as memory full, task failed etc.

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

Mode: Administrator execution

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

Displaying reports

The **show reports** command is used to dump combined system information. The **show reports** command sequentially executes the following log commands:

```
show version
show clock
show uptime
show licenses
show memory stat
show log reset
show log boot
show log event
show log supervisor
show factory-config
show startup-config
show running-config
```

Mode: Administrator execution

Step	Command	Purpose
1	***node*#show reports**	Dumps the combined system information.

Controlling command execution

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

Mode: Administrator execution

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

Example: Controlling Command Execution

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

```
node>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

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

Show the currently running commands

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

Bring job 0 to foreground

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

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

Mode: Administrator execution

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

Example: Displaying the checksum of a configuration

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

```
node#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.

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

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

Chapter 8 RADIUS Client Configuration

Chapter contents

Introduction 103
The AAA component 103
General AAA Configuration 104
RADIUS configuration 106
Configuring RADIUS clients 107
Configuring RADIUS accounting 108
Configuring the RADIUS server 110
Attributes in the RADIUS request message 110
Attributes in the RADIUS accept message 111
Configuring the local database accounts 111
Storing call logs with quality information 113

Introduction

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

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

This chapter includes the following sections:

- The AAA component
- RADIUS configuration (see page 106)
- Configuration of the local database accounts (see page 111)

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).

Figure 14 illustrates the authentication procedure for a user logging into a SmartNode that is configured to use RADIUS as authentication method.

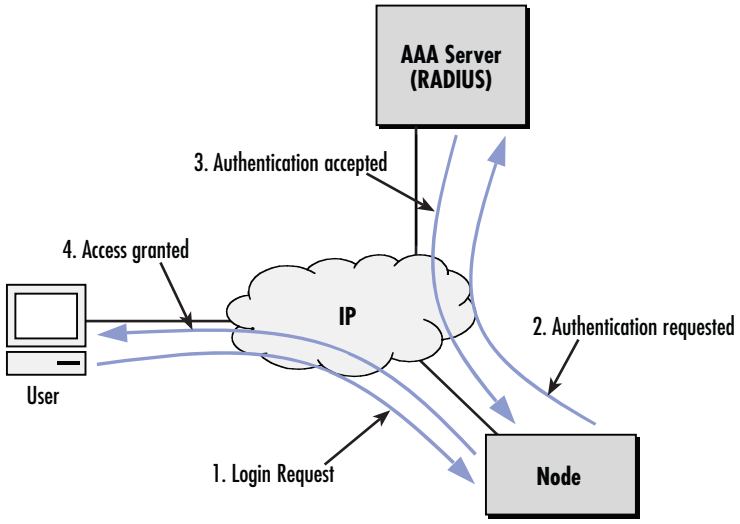


Figure 14. Authentication procedure with a RADIUS server

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

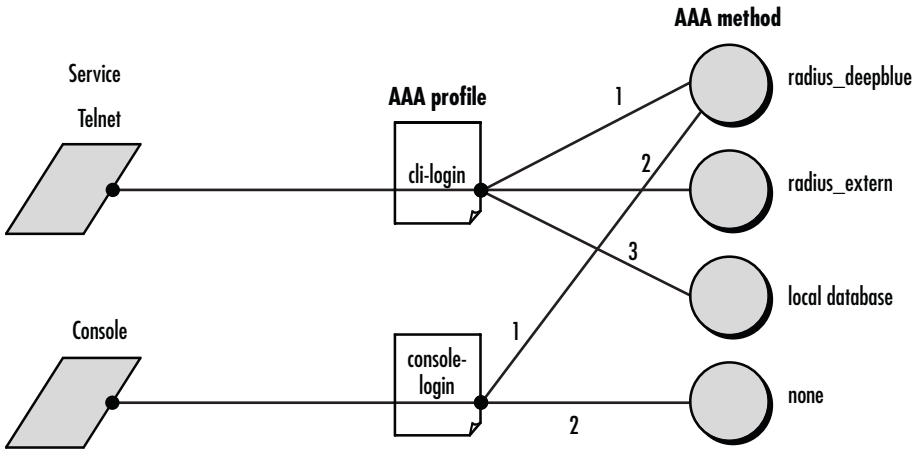


Figure 15. How to use AAA methods and AAA profiles

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

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

3. Query the local database (see "Configuring the local database accounts" on page 111 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 authentication** *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 9, "IP context overview" on page 114. 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 authentication** *profile-name*	Specifies which AAA profile the Telnet login service has to use.
7	***node*(cfg)#terminal console use authentication** *profile-name*	Specifies which AAA profile the console login service has to use.
8	***node*(cfg)#show profile authentication** [*name*]	Displays the configured profiles

Example: Create the AAA profiles for login over Telnet and login over console, as they are shown in figure 15, and use them on the Telnet login and console login services.

```
node>enable
node#configure
node(cfg)#profile authentication remote-radius
node(pf-auth)[remote-~]#method radius radius_deepblue
node(pf-auth)[remote-~]#method radius radius_extern
node(pf-auth)[remote-~]#method local
node(pf-auth)[remote-~]#server-timeout 15
```

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

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

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

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

node(cfg)#
```

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 74.

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 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 15.

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

```
node(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. For more information about each attribute, or other possible attributes, see RFC 2865 or the documentation of the radius server you use.

Attributes in the RADIUS request message

The SmartNode sends a RADIUS request with the following attributes:

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

Attributes in the RADIUS accept message

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

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

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

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

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

Type: 26

Length: Length of the whole attribute including the vendor data

Vendor-Id: 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 SmartWare is to set up local user accounts. The local database—which is queried with the AAA method *local* as described previously—can contain administrator and operator accounts. For example, to grant access to the local SmartNode if all RADIUS

servers are down or the network is not reachable, you can create an emergency user in the local database so that you can still access the SmartNode. Perform the following steps to configure the local accounts.

Mode: Configure

Step	Command	Purpose
1	**node(cfg)#[no] administrator** *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 SmartNode) and the remote RADIUS server.
4	**node(pf-auth)[name]#show accounts**	Display existing accounts

Example: Create an administrator and an operator account

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

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

Storing call logs with quality information

It is possible to store call logs with quality information in the local aaa data sink. To do so, a service aaa is needed where the calls are routed through. Create a profile aaa with the method local. Then create a service aaa to route the calls through.

Example:

```
profile aaa default
  method 1 local

context cs switch

  routing-table called-e164 TAB_OUT
    route 200 dest-interface IF_SIP

  interface isdn IF_BRI_00
    route call dest-service QoS_LOG.OUT.bri

  interface sip IF_SIP
    bind context sip-gateway sip
    route call dest-service QoS_LOG.OUT.ethernet

  service aaa QoS_LOG.OUT
    accounting use profile aaa LOCAL_AAA

    port bri
      route call dest-table TAB_OUT

    port ethernet
      route call dest-interface IF_BRI_00
```

To see the collected logs use the command **`show accounting`** or use the Web-GUI to see the logs. In the Web-GUI, go to the '**Reports**' page and select "**Call Quality Log**" in the tab bar on top of the page. On that page, ou will also have the option to export the log as a comma separated text file.

Chapter 9 IP context overview

Chapter contents

Introduction 115
IP context overview configuration task list 116
Planning your IP configuration 117
IP interface related information 117
QoS related information 117
Configuring physical ports 117
Creating and configuring IP interfaces 117
Configuring NAPT 118
Configuring static IP routing 118
Configuring RIP 118
Configuring access control lists 119
Configuring quality of service (QoS) 119

Introduction

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

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

To understand the information given in the following chapters, carefully read to the end of the current chapter. Before proceeding, make sure that you feel comfortable with the underlying SmartWare configuration concept by reading chapter 2, "Configuration concepts" on page 44.

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

In figure 16 on page 115, the IP context with all its related elements is contained within the area on the left, which has a gray fill. The right side displays the related CS context, which communicates with the IP context via different types of gateways. Since the CS context and its related components are not the subject of this chapter, they are illustrated in figure 16 with gray lines instead of black ones.

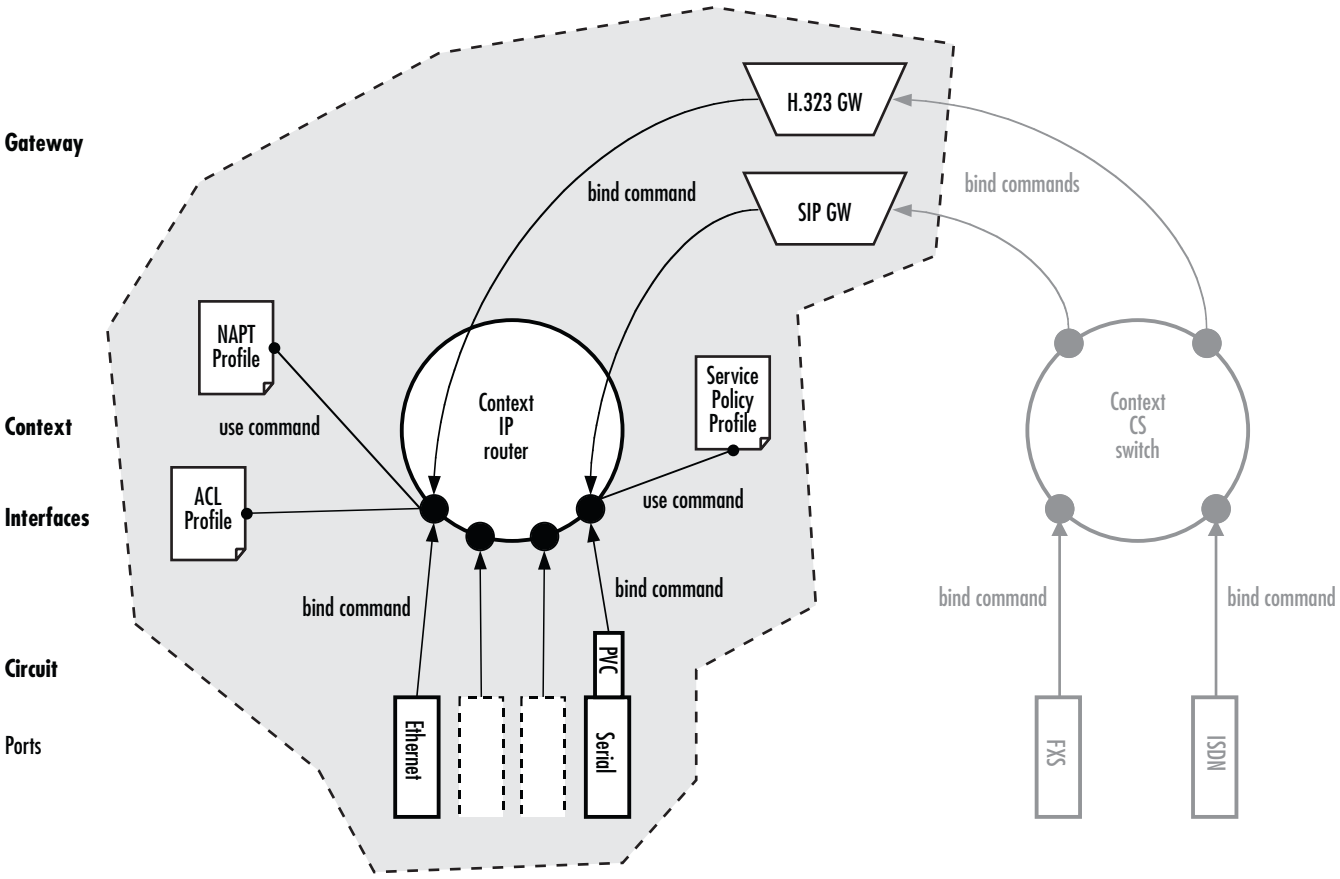


Figure 16. IP context and related elements

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 10, "IP interface configuration" on page 120.
- You can find the information regarding network address port translation (NAPT) in chapter 11, "NAT/NAPT configuration" on page 132.
- If you need to configure a physical port, chapter 12, "Ethernet port configuration" on page 141 or chapter 14, "Serial port configuration" on page 170 may prove helpful.
- To set up the IP router contained in SmartWare, chapter 22, "Basic IP routing configuration" on page 235 and chapter 23, "RIP configuration" on page 242 give you the required information.
- For essential knowledge related to network security requirements, refer to chapter 24, "Access control list configuration" on page 253.
- If your network shall provide better service to selected network traffic, chapter 13, "Link scheduler configuration" on page 151 will help you to get in-depth knowledge about quality of service (QoS) management with SmartWare.

The following sections describe the basic tasks involved in IP context configuration. Many parameters have acceptable default values, which in most cases do not need to be explicitly configured. Hence not all of the configuration tasks below are required. Depending on your application scenario, some tasks are mandatory or might be optional. The following tasks use a bottom-up approach, starting from the ports, followed by the interfaces up to the services running on the SmartNode. The first tasks below shall help you obtaining the necessary overview, in view of the fact that there is always a risk getting lost in details before gaining a general understanding of the whole network.

- Planning your IP configuration (see page 117)
- Configuring Ethernet and serial ports (see page 117)
- Creating and configuring IP interfaces (see page 117)
- Configuring NAPT (see page 118)
- Configuring static IP routing (see page 118)
- Configuring RIP (see page 118)
- Configuring access control lists (see page 119)
- Configuring quality of service (see page 119)

Planning your IP configuration

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

IP interface related information

Setting up the basic IP connectivity for your SmartNode requires the following information:

- IP addresses used for Ethernet LAN and WAN ports
- IP Subnet mask used for Ethernet LAN and WAN ports
- Length for Ethernet cables
- IP addresses of the central H.323 gatekeeper or SIP registrar
- IP addresses of the central PSTN gateway for H.323 and SIP based calls
- IP address of the central TFTP server used for configuration upload and download

QoS related information

Check with your access service provider if there are any QoS related requirements, which you need to know prior to configuring SmartWare QoS management. Check the following with your access service provider:

- What is the dedicated bandwidth, which you have agreed with your access service provider?
- How does your provider perform packet classification, e.g. which ToS bits have to be used to define the supported classes of service?

Configuring physical ports

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 ports, refer to the respective port type's chapter.

Creating and configuring IP interfaces

The number and names of IP interfaces depend upon your application scenario. An interface is a logical construct that 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.

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

For information and examples on how to create and configure an IP interface, refer to chapter 10, "IP interface configuration" on page 120.

Configuring NAPT

Network address port translation (NAPT), which is an extension to NAT, uses TCP/UDP ports in addition to network addresses (IP addresses) to map multiple private network addresses to a single outside address. NAPT enables small offices to save money by requiring only one official outside IP address to connect several hosts via a SmartNode to the access network. Moreover, NAPT provides additional security, because the IP addresses of hosts attached via the SmartNode are invisible to the external world. You can configure NAPT by creating a profile that is afterwards used on an explicit IP interface. In SmartWare terminology, an IP interface *uses* a NAPT profile, as shown in figure 16 on page 115.

For information and examples on how to configure NAPT refer to chapter 11, "NAT/NAPT configuration" on page 132.

Configuring static IP routing

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

For information and examples on how to configure static IP routing, refer to chapter 22, "Basic IP routing configuration" on page 235.

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 23, "RIP configuration" on page 242.

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.

An access control list is a sequential collection of permit and deny conditions that apply to packets on a certain interface. Access control lists can be configured for all routed network protocols (IP, ICMP, TCP, UDP, and SCTP) to filter the packets of those protocols as the packets pass through a SmartNode. SmartWare tests packets against the conditions in an access list one by one. The first match determines whether SmartWare accepts or rejects the packet. Because SmartWare stops testing conditions after the first match, the order of the conditions is critical. If no conditions match, the software rejects the address.

For information and examples on how configure access control lists, refer to chapter 24, "Access control list configuration" on page 253.

Configuring quality of service (QoS)

The link scheduler enables the definition of QoS profiles for network traffic on a certain interface, as shown in figure 16 on page 115. QoS refers to the ability of a network to provide improved service to selected network traffic over various underlying technologies including Frame Relay, Ethernet and 802.x type networks, and IP-routed networks. In particular, QoS features provide improved and more predictable network service by providing the following services:

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

The QoS features described in chapter 13, "Link scheduler configuration" on page 151 address these diverse and common needs.

Chapter 10 IP interface configuration

Chapter contents

Introduction 121
IP interface configuration task list 121
Creating an IP interface 121
Deleting an IP interface 122
Setting the IP address and netmask 123
Configuring a NAPT DMZ interface 123
ICMP message processing 124
ICMP redirect messages 124
Router advertisement broadcast message 124
Defining the MTU and MSS of the interface 125
Configuring an interface as a point-to-point link 126
Displaying IP interface information 126
Displaying dynamic ARP entries 127
Flushing dynamic ARP entries 127
Processing gratuitous ARP requests 127
Testing connections with the ping command 127
IP link supervision 128
Check connectivity of an IP link 129
Show IP link status 129
Debug connectivity 129
Debug ARP 129
Traceroute 130
Configuring the IGMP Proxy 131

Introduction

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

An interface is a logical entity that provides higher-layer protocol and service information, such as Layer 3 addressing. Interfaces are configured as part of a context and are independent of physical ports and circuits. The separation of the interface from the physical layer allows for many advanced features. For higher layer protocols to become active, a physical port or circuit must be bound to an interface. IP interfaces can be bound physically to Ethernet, SDSL or Frame Relay ports according to the appropriate transport network layer.

IP interface configuration task list

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

- Creating an IP interface (see page 121)
- Deleting an IP interface (see page 122)
- Setting the IP address and netmask (see page 123)
- ICMP message processing (see page 124)
- ICMP redirect messages (see page 124)
- Router advertisement broadcast message (see page 124)
- Defining the MTU of the interface (see page 125)
- Configuring an interface as a point-to-point link (see page 126)
- Displaying IP interface information (see page 126)
- Testing connections with the **ping** command (see page 127)

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.

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

```
node(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:

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

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

```
node(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.
dhcp	Enables the DHCP client on this interface. For more information on DHCP-client configuration refer to chapter 27, "DHCP configuration" on page 292.

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.

```
node(ctx-ip)[router]#interface lan
node(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. SmartWare supports the following ICMP message processing features:

- ICMP redirect messages
- Router advertisement broadcast message

ICMP redirect messages

Routes are sometimes less than optimal. For example, the router may be forced to resend a packet through the same interface on which it was received. In this case, an ICMP redirect message is sent to the originator of the packet telling 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.

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.

```
node(ctx-ip)[router]#interface lan
node(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.

```
node(ctx-ip)[router]#interface lan
node(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 IP packet that exceeds the MTU set for an interface is exceeded. 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.

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

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

Displaying IP interface information

The **show ip interface** command 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.

```
node>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.

Processing gratuitous ARP requests

The following command can be used to accept and process gratuitous ARP requests and replies. ARP requests and replies in which the target protocol address and sender protocol address are the same are considered. Because of security reasons, this feature is deactivated as default and must be enabled by the user.

Mode: configure

Step	Command	Purpose
1	[*name*]**(cfg)#[no] arp gratuitous**	Enables or disables the processing of gratuitous arp requests and replies.

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 <address> [<number>] [timeout <seconds>] [packet-size <packet-size>] [ttl <ttl>] [traffic-class <traffic-class>]***	Sends ICMP ECHO_REQUEST packets to network hosts at IP address <address>

Where the parameters are defined as follows:

- [**<number>**] optional parameter which indicates how many pings are sent
- [**timeout <seconds>**] optional parameter which indicates the time-out period of the ping
- [**packet-size <packet-size>**] optional parameter which indicates the number of octets in the ping
- [**ttl <ttl>**] optional parameter which indicates the time-to-live value
- [**traffic-class <traffic-class>**] which indicates the IP packets in a traffic class are routed via the defined Nexthop in the routing table entry for that traffic-class. Default: local-default.

When using **ping** for fault isolation, you should first run it on the respective IP 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.

```
node>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
```

IP link supervision

IP Link Supervision is one of the modules you have to configure in order to use the PPP dial-up over ISDN feature. Also consider the **dial-up** command (page 337) on the IP interface and the **interface dialer** mode (page 331) in context cs.

IP link supervision can be used to periodically check the reachability of some hosts over a specific link. Therefore an ICMP echo request is sent to the configured IP address. After a configurable number of failed requests the host is considered unreachable. If all configured hosts are unreachable, the interface will be closed for nor-

mal traffic and the IP router removes all routes pointing to that interface. Now, traffic previously flowing over that interface is routed through the interface with IP routes of a higher metric. However the interface continues to send ICMP echo requests. After a configurable number of ICMP replies the host is reachable again, and the interface will be opened for normal traffic.

Check connectivity of an IP link
Mode: context ip/interface

Step	Command	Purpose
1	[*name*] **(if-ip)** [*interface*]**#[no] check-connectivity ping <ip-address> [source-address <ip-address>] [tolerance-down <number>] [tolerance-up <number>] [interval <seconds>] [timeout <seconds>]**	Configures or removes a host to ping. Source-address specifies the IP source address for the ICMP packets. Tolerance-down specifies the number of pings allowed to fail (Default 3). Tolerance-up specifies the number of successful pings required to activate link (Default 1). Interval specifies the interval in which the pings are sent (Default 10). Timeout specifies the time in seconds to wait for an answer (Default 3).

Show IP link status
The following command shows the status of connectivity.

Mode: enable

Step	Command	Purpose
1	[*name*]**#show ip connectivity [<interface>]**	Shows the status of connectivity.

Debug connectivity
The following command enables logging of connectivity events.

Mode: enable

Step	Command	Purpose
1	[*name*]**#debug connectivity**	Enables logging of connectivity events and state changes.

Debug ARP

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

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.

Traceroute

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

Step	Command	Purpose
1	***node#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] [traffic-class <traffic-class>]***	Prints the route that the packets take to the network host. Optionally, a traffic-class can be specified in the 'traceroute' command. 'traceroute' follows the route of the specified traffic-class. Default: local-default

Example: Debug ARP output

```
node(cfg)#debug arp
node(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.

```
node(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.

Chapter 11 NAT/NAPT configuration

Chapter contents

Introduction 133
Dynamic NAPT 133
Static NAPT 134
Dynamic NAT 134
Static NAT 135
NAPT traversal 135
NAT/NAPT configuration task list 136
Creating a NAPT profile 136
Configuring a NAPT DMZ host 137
Defining NAPT port ranges 137
Preserving TCP/UDP port numbers in NAPT 138
Defining the UDP NAPT type 138
Activate NAT/NAPT 139
Displaying NAT/NAPT configuration information 139
Configuring NAT static protocol entries 140

Introduction

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

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

SmartWare provides four types of NAT/NAPT:

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

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

Dynamic NAPT

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

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

Figure 17 illustrates the basic and enhanced behavior of the Dynamic NAPT. The big arrows indicate the direction of the connection establishment. Although only a local host can establish a connection, traffic always flows in both directions.

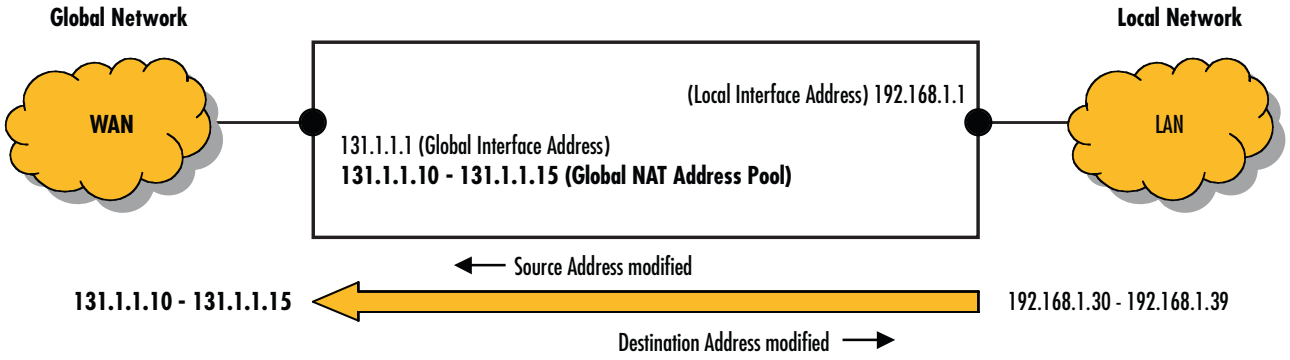


Figure 17. Dynamic NAPT

Static NAPT

Dynamic NAPT does not permit hosts on the global network to access hosts on the local network. Static NAPT makes selected services (i.e. ports) of local hosts globally accessible. Static NAPT entries map global addresses/ports to local addresses/ports. The global address can either be the address of the global interface or a configured global NAPT address. Usually, the local and the global port of a static NAPT entry are the same; however, they may be different.

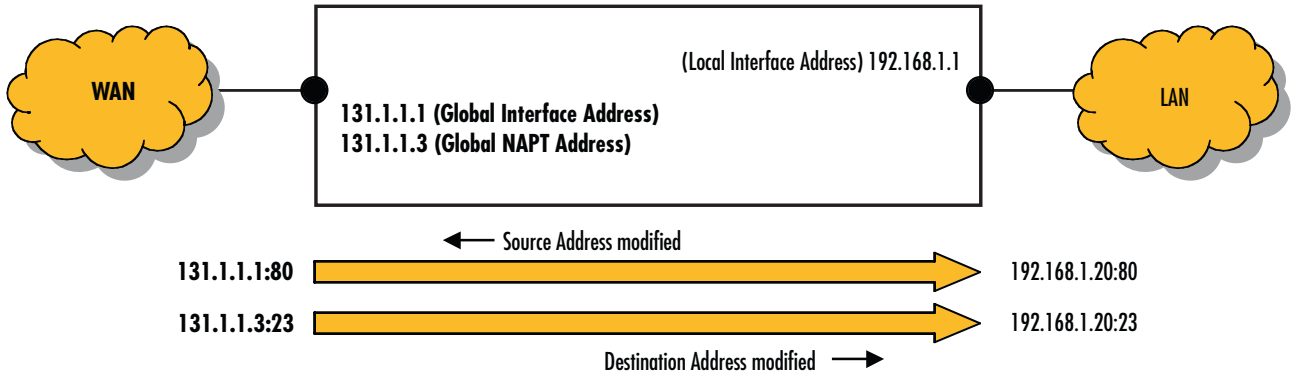


Figure 18. Static NAPT

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

Dynamic NAT

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

Dynamic NAT is particularly useful for protocols that do not build on UDP or TCP but directly on IP (e.g. GRE, ESP). See also section "NAPT traversal" on page 135.

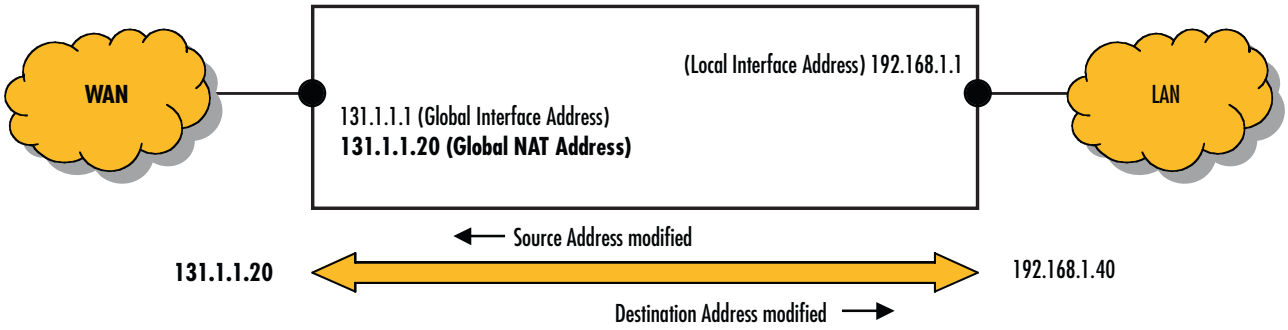


Figure 19. Dynamic NAT

Static NAT

Dynamic NAT does not permit hosts on the global network to access hosts on the local network. Static NAT makes local hosts globally accessible. Static NAT entries map global addresses to local addresses. The global address must be a configured global NAT address. It cannot be the address of the global interface since this would break connectivity to the SmartNode itself.

Static NAT is particularly useful for protocols that do not build on UDP or TCP but directly on IP (e.g. GRE, ESP). See also section "NAPT traversal" on page 135.

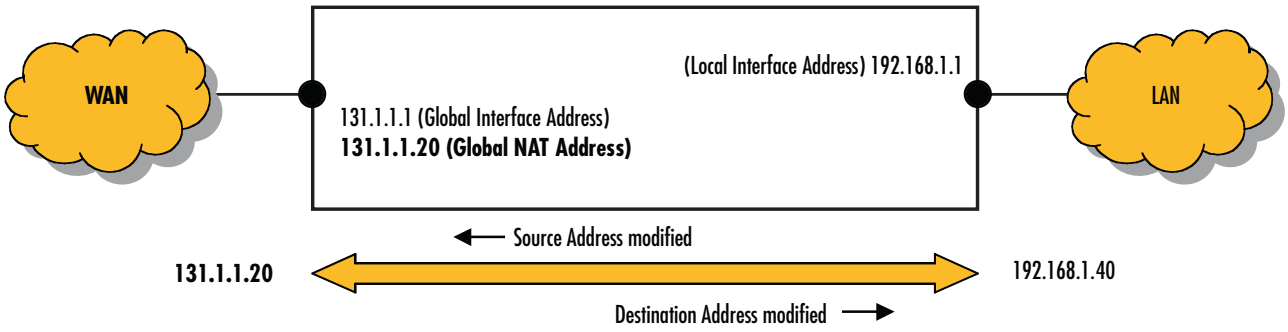


Figure 20. Static NAT

NAPT traversal

Protocols that do not build on UDP or TCP but directly on IP (e.g. GRE, ESP), and protocols that open additional connections unknown to the NAT/NAPT component (e.g. FTP, H.323, SIP), do not easily traverse a NAPT.

The SmartWare NAPT can handle one GRE (Generic Routing Encapsulation) connection and one ESP (Encapsulating Security Payload) connection at a time. It also routes ICMP messages back to the source of the concerned connection or to the source of an ICMP Ping message.

To enable NAPT traversal of protocols that open additional connections, the NAPT component must analyze these protocols at the Application Level in order to understand which NAPT entries for additional connections

it should create and which IP addresses/ports it must modify (e.g. for voice connections in addition to signaling connections). It performs this task for the protocol FTP. Other protocols such as H.323 and SIP cannot traverse the SmartWare NAPT.

NAT/NAPT configuration task list

To configure the NAT/NAPT component, perform the tasks in the following sections:

- Creating a NAPT profile (see page 136)
- Activating NAT/NAPT (see page 136)
- Displaying NAT/NAPT configuration information (see page 139)

Creating a NAPT profile

A NAPT profile defines the behavior of the NAT/NAPT component, comprising all four types of NAT/NAPT (this profile is called 'NAPT profile' and not 'NAT/NAPT profile for historical reasons). Several NAPT profiles are admissible but there is only one NAT/NAPT component.

Procedure: To create a NAPT profile and to configure the required types of NAT/NAPT

Mode: Configure

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)
4 (optional)	*node*(pf-napt)[*name*]#**range** *local-ip-range-start local-ip-range-stop global-ip-start global-ip-stop*	Configures and activates the Dynamic NAT: *local-ip-range-start* and *local-ip-range-stop* define the subset of local hosts that use an address from the global NAT address pool to access to global network. *global-ip-start* and *global-ip -stop* define the global NAT address pool. (max. 20 entries) The IP ranges of different Dynamic NAT entries must not overlap each other.

Step	Command	Purpose
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 133.

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

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

```
node(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 12 **Ethernet port configuration**

Chapter contents

Introduction 142
Ethernet port configuration task list 142
Entering the Ethernet port configuration mode 142
Configuring medium for an Ethernet port 142
Configuring Ethernet encapsulation type for an Ethernet port 143
Binding an Ethernet port to an IP interface 143
Multiple IP addresses on Ethernet ports 144
Configuring a VLAN 145
Configuring layer 2 CoS to service-class mapping for an Ethernet port 146
Adding a receive mapping table entry 147
Adding a transmit mapping table entry 148
Closing an Ethernet port 148
Using the built-in Ethernet sniffer 149

Introduction

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

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 142)
- Configuring medium for an Ethernet port (see page 142)
- Configuring Ethernet encapsulation type for an Ethernet port (see page 143)
- Binding an Ethernet port to an IP interface (see page 143)
- Configuring multiple IP addresses on the Ethernet ports (see page 144)
- Configuring a VLAN (see page 145)
- Configuring layer 2 CoS to service-class mapping for an Ethernet port (advanced) (see page 146)
- Closing an Ethernet port (see page 148)

Entering the Ethernet port configuration mode

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

Configuring medium for an Ethernet port

All Ethernet ports are configured by default to auto-sense both the port speed and the duplex mode. This is the recommended configuration. Command options are (if supported by the platform):

- **10**—for 10 Mbps
- **100**—for 100 Mbps
- **1000**—for Gigabit Ethernet
- **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 | 1000 | auto} (half | full)**	Configures the interface on *slot* and *port* to medium according to the selected option.

Example: Configuring medium for an Ethernet port

The following example shows how to configure medium auto-sense for the Ethernet port on slot 0 and port 0 of a SmartNode 4524 device.

```
node(cfg)#port ethernet 0 0
node(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.

```
node(cfg)#port ethernet 0 0
node(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 21 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.

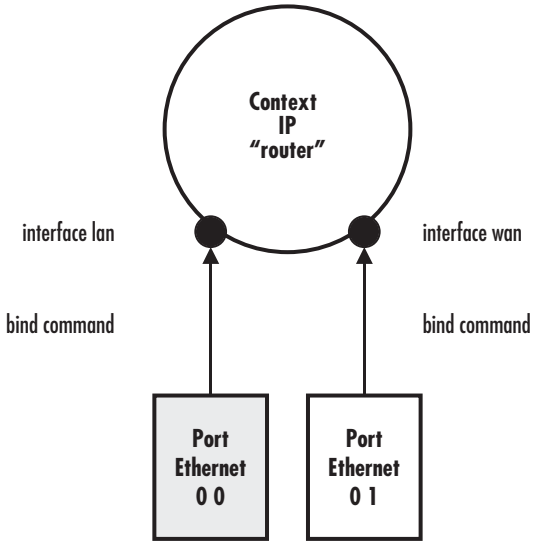


Figure 21. Binding of an Ethernet port to an IP interface

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* to an already existing IP interface *lan*.

```
node(cfg)#port ethernet 0 0
node(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 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 30, "PPP configuration" on page 313.
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 a SmartNode.

```
node(cfg)#port ethernet 0 0
node(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.

```
node(cfg)#port ethernet 0 0
node(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.

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

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

```
node(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.

```
node(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 ethernet 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 13 Link scheduler configuration

Chapter contents

Introduction 152
Applying scheduling at the bottleneck 152
Using traffic classes 152
Introduction to Scheduling 153
Priority 153
Weighted fair queuing (WFQ) 153
Shaping 153
Burst tolerant shaping or wfq 154
Hierarchy 154
Quick references 155
Setting the modem rate 155
Command cross reference 156
Link scheduler configuration task list 156
Defining the access control list profile 157
Packet classification 157
Creating an access control list 158
Creating a service policy profile 159
Specifying the handling of traffic-classes 161
Defining fair queuing weight 161
Defining the bit-rate 162
Defining absolute priority 162
Defining the maximum queue length 162
Specifying the type-of-service (TOS) field 162
Specifying the precedence field 163
Specifying differentiated services codepoint (DSCP) marking 163
Specifying layer 2 marking 164
Defining random early detection 165
Discarding Excess Load 165
Quality of Service for routed RTP streams 165
Devoting the service policy profile to an interface 167
Displaying link arbitration status 168
Displaying link scheduling profile information 168
Enable statistics gathering 168

Introduction

This chapter describes how to use and configure the Quality of Service (QoS) features. Refer to chapter 24, "Access control list configuration" on page 253 more information on the use of access control lists.

This chapter includes the following sections:

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

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

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

Applying scheduling at the bottleneck

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

Using traffic classes

The link scheduler needs to distinguish between different types of packets. We refer to those types as "traffic-classes". You can think of the traffic-class as if every packet in the SmartNode has a tag attached to it on which the classification can be noted. The access control list "stage" (ACL) can be used to apply such a traffic-class name to some type of packet based on its IP-header filtering capabilities. The traffic-class tags exist only inside the SmartNode, but layer 2 priority bits (802.1pq class-of-service) and IP header type-of-service bits (TOS field) can be used to mark a specific packet type for the other network nodes. By default the traffic-class tag is empty. Only two types of packets are automatically marked by the SmartWare: voice packets and data packets origination from or destined to the SmartNode itself are marked as "local-voice" and "local-default" respectively. Please refer to figure 22 on page 153 when using the ACL to classify traffic. It illustrates the sequence of processing stages every routed packet passes. Only stages that have been installed in the data path with a "use profile..." statement in the corresponding interface configuration are present. Both an input direction ACL on the receiving interface as well as an output ACL on the transmitting interface can be used to classify a packet for special handling by the output link scheduler on the transmit interface. But as visible from the figure no ACL can be used for an input link scheduler.

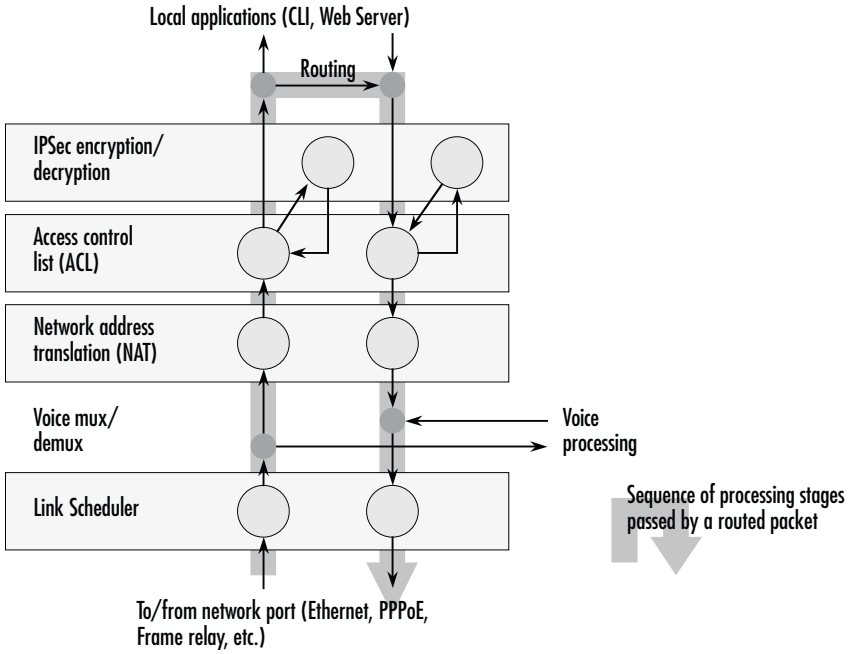


Figure 22. Packet routing in SmartWare

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

Introduction to Scheduling

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

Priority

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

Weighted fair queuing (WFQ)

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

Shaping

There is another commonly used way to assign bandwidth. It is called *shaping* and it makes sure that each traffic-class will get just as much bandwidth as configured and not more. This is useful if you have subscribed to a

service that is only available for a limited bandwidth e.g. low delay. When connecting the SmartNode to a *Diff-Serv* network shaping might be a required operation.

Burst tolerant shaping or wfq

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

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

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

Hierarchy

An arbiter can either use wfq *or* shaping to determine which source to serve next. If you want the scheduler to follow a combination of decision criteria you can combine different schedulers in hierarchy to do a multi-level arbitration.Hierarchical scheduling is supported in SmartWare with service-policy profiles used inside service-policy profiles.In figure 23 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.

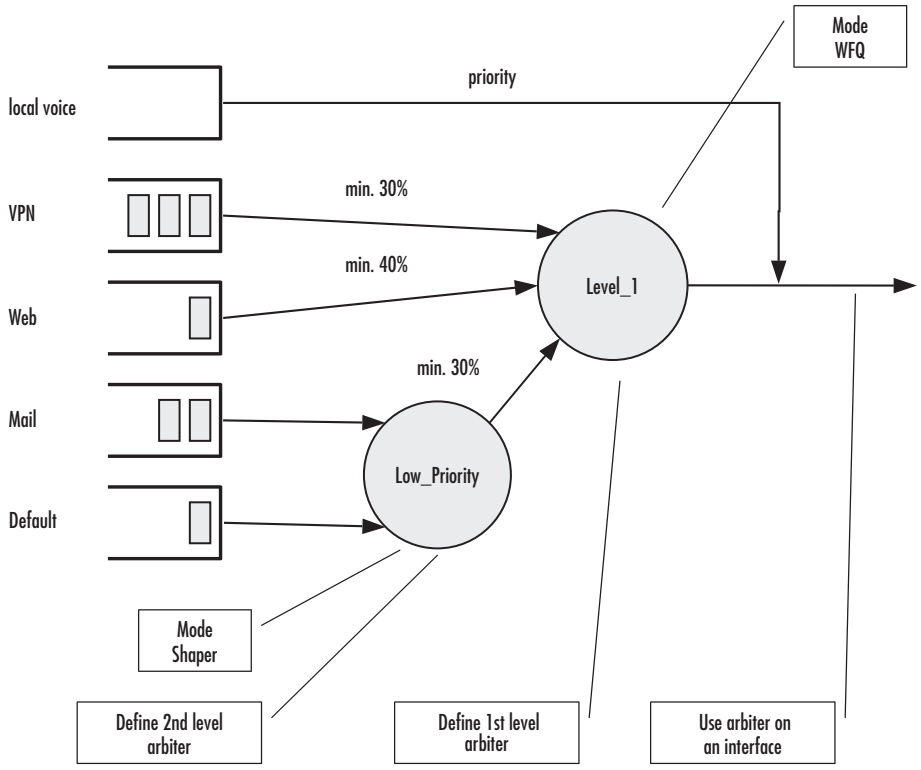


Figure 23. Example of Hierarchical Scheduling

Quick references

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

Setting the modem rate

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

1. Create a minimal profile.

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

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

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

Some explanations:

- "modem-512" is the title of the profile which is referred to when installing the scheduler

- "rate-limit 512" allows no more than 512 kbit/sec to pass which avoids 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 SmartWare with the Cisco IOS QoS software command syntax often helps administrators to straightforwardly configure SmartNode devices. In table 4 the Cisco IOS Release 12.2 QoS commands are in contrast with the respective SmartWare commands.

Table 4. Command cross reference

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

Link scheduler configuration task list

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

- Defining the access control list profile
- Creating a service-policy profile (see page 159)
- Specifying the handling of traffic-classes (see page 161)
- Devoting the service policy profile to an interface (see page 167)
- Displaying link arbitration status (see page 168)
- Displaying link scheduling profile information (see page 168)

- Enable statistics gathering (see page 168)

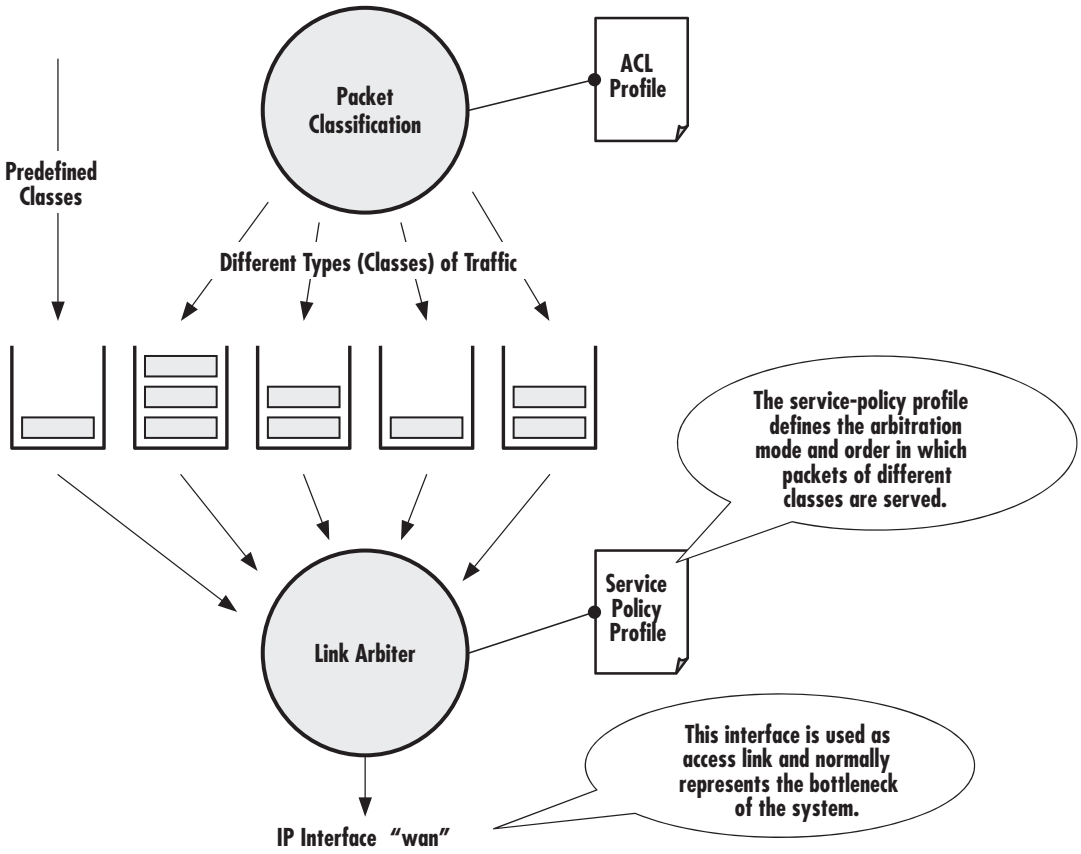


Figure 24. Elements of link scheduler configuration

Defining the access control list profile

Packet classification

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

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

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

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

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

Creating an access control list

The procedure to create an access control list is described in detail in chapter 24, "Access control list configuration" on page 253.

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 25. 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.

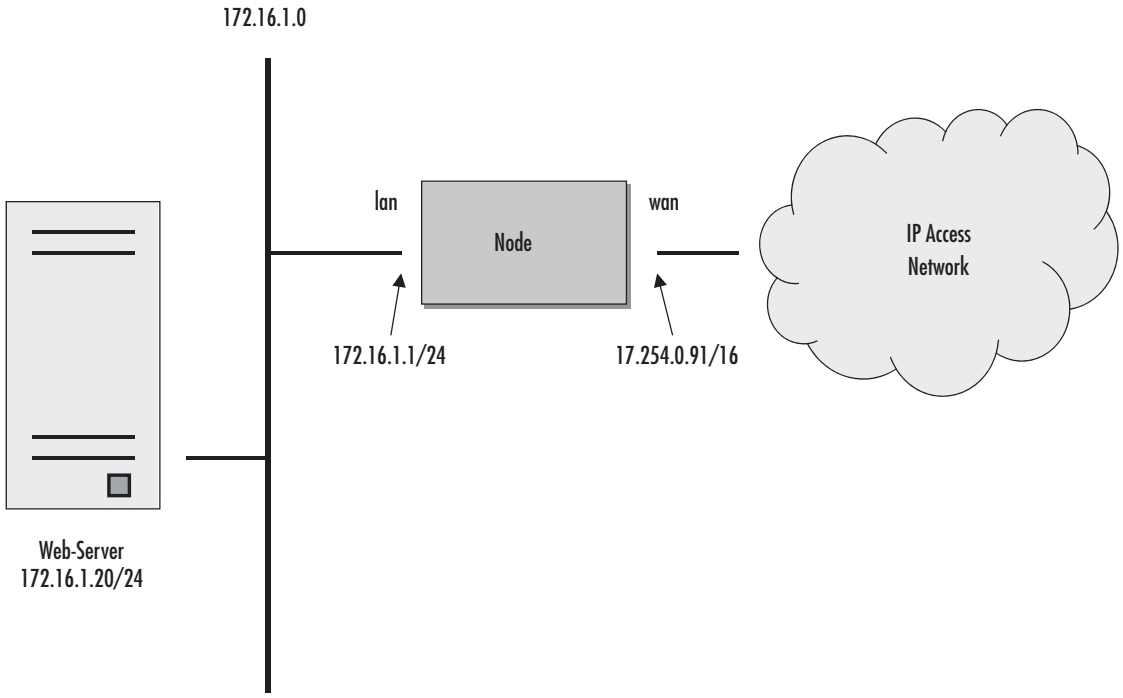


Figure 25. Scenario with Web server regarded as a single source host

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.

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

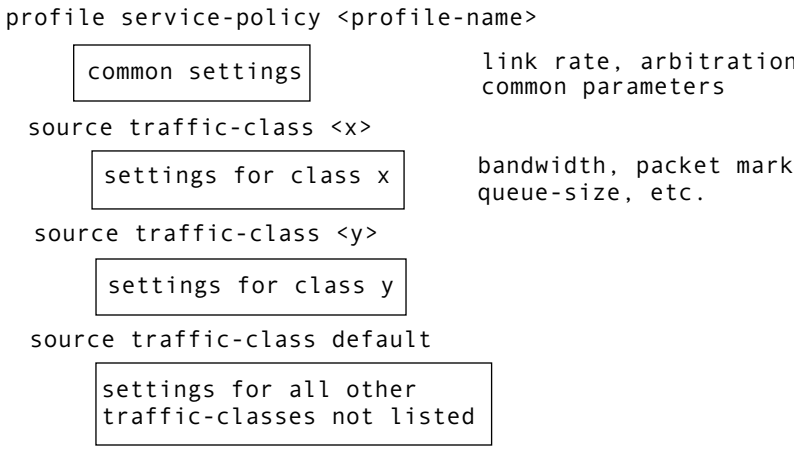


Figure 26. Structure of a Service-Policy Profile

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

Example: Creating a service policy profile

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

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

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

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

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

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

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

Mode: Configure

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

Specifying the handling of traffic-classes

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

Defining fair queuing weight

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

Mode: Source

Command	Purpose
***node*(src)[*name*]#share** *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 5. TOS values and their meaning

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

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

Mode: Source

Command	Purpose
***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 5 on page 163, but any number from 0 to 15 can be configured.

Specifying the precedence field

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

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

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

Mode: Source

Command	Purpose
***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 SmartWare you can inspect the DSCP value in the ACL rules and modify the DSCP value with the link scheduler **set ip dscp** command.

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

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

Mode: Source

Command	Purpose
***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 6 lists the tag components.

Table 6. 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

SmartWare 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 SmartWare a service policy profile is used on a certain IP interface, as shown in figure 27.

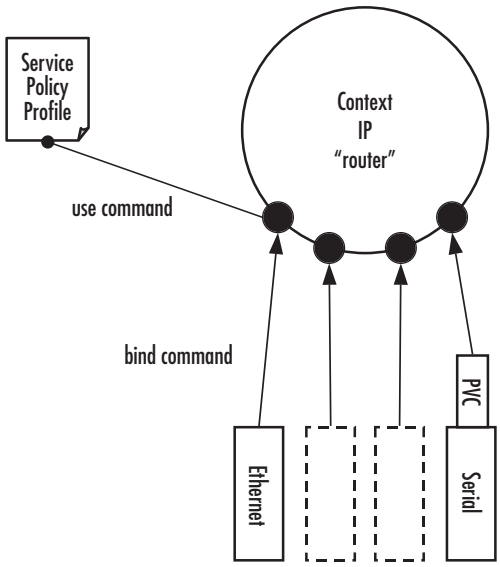


Figure 27. Using a Service Policy Profile on an IP Interface

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.

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

```
node(ctx-ip)[router]#interface wan
node(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.

```
node>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*.

```
node#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 7).

Table 7. 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 SmartWare require the CPU resources of your SmartNode. Therefore do not enable statistic gathering or other debug features if it is not necessary. Disable any debug feature after use with the **no** form of the command.

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

Mode: Source

Step	Command	Purpose
1	***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

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

Chapter 14 Serial port configuration

Chapter contents

Introduction 171
Serial port configuration task list 171
Disabling an interface 171
Enabling an interface 172
Configuring the serial encapsulation type 173
Configuring the hardware port protocol 173
Configuring the active clock edge 174
Configuring the baudrate 175

Introduction

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

- Serial port configuration task list
- Configuration tasks
- Examples

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

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

Serial port configuration task list

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

- Disabling an interface (see page 171)
- Enabling an interface (see page 172)
- Configuring the serial encapsulation type (see page 173)
- Configuring the hardware port protocol (see page 173)
- Configuring the active clock edge (see page 174)
- Configuring the baudrate

Disabling an interface

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

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

This procedure describes how to shut down a serial interface

Mode: Administrator execution

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

Example: Disabling an interface

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

```
node(cfg)#port serial 0 0
node(prt-ser)[0/0]#shutdown
node(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, 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. Check that *State* is set to *OPENED* in the command output of **show port serial**.

```
node(cfg)#port serial 0 0
node(prt-ser)[0/0]#no shutdown
node(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 and PPP serial encapsulation method. For more information how to configure Frame Relay and PPP, please see Chapter15, "Frame Relay configuration" on page 177 and Chapter 30, "PPP configuration" on page 313.

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.

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. Check that in the command output of **show port serial** *Encapsulation* is set to *framerelay*.

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

Serial Interface Configuration
------------------------------

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

Configuring the hardware port protocol

Note Only available on certain devices.

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

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

Mode: Administrator execution

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

Example: Configuring the hardware port protocol

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

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

Serial Interface Configuration
------------------------------

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

Configuring the active clock edge

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

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. Check that *Transmit Clock* is set to *inverted* in the command output of **show port serial**.

```
node(cfg)#port serial 0 0
node(prt-ser)[0/0]#transmit-data-on-edge negative
node(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 desired baudrate can be configured.

Note Only available on certain devices.

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.

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

Chapter 15 **Frame Relay configuration**

Chapter contents

Introduction 178
Frame Relay configuration task list 178
- Configuring Frame Relay encapsulation 178
- Configuring the LMI type 179
- Configuring the keep-alive interval 179
- Enabling fragmentation 180
- Entering Frame Relay PVC configuration mode 181
- Configuring the PVC encapsulation type 182
- Binding the Frame Relay PVC to IP interface 182
- Enabling a Frame Relay PVC 184
- Disabling a Frame Relay PVC 184
- Debugging Frame Relay 185
- Displaying Frame Relay information 186
- Integrated service access 187
- Example 1: Frame Relay on e1t1 without a channel-group 189

Introduction

This chapter provides an overview of the Frame Relay protocol and describes the tasks involved in its configuration includes the following sections:

- Frame Relay configuration task list
- Configuration tasks
- Examples

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.

Frame Relay configuration task list

Perform the tasks in the following sections to configure Frame Real on various ports:

- Configuring Frame Relay encapsulation
- Configuring the LMI type
- Configuring the keep-alive interval
- Enabling fragmentation
- Entering Frame Relay PVC configuration mode
- Configuring the PVC encapsulation type
- Binding the Frame Relay PVC to IP interface
- Disabling a Frame Relay PVC
- Displaying Frame Relay information

Configuring Frame Relay encapsulation

Normally, Frame Relay is used over a HDLC framed link. Different kind of physical ports can be configured for HDLC framed data transmission. On some ports the hdlc mode must be explicitly enabled (PRI, BRI), other ports have a HDLC framed nature (Serial). That means, Frame Relay encapsulation can be configured in different configuration modes. For this reason, the command description below refers to the configuration mode in which Frame Relay can be enabled by setting the encapsulation to 'framerelay'. This configuration mode is called here 'hdlc-sub' but it is only an alias for the real mode. Once encapsulation framerelay has been configured, the Frame Relay configuration mode can be entered.

Mode: hdlc-sub

Step	Command	Purpose
1	***node*(hdlc-sub)#[no] encapsulation framerelay**	Enables/Disables Frame Relay
2	***node*(hdlc-sub)#framerelay**	Enters the framerelay configuration mode

Configuring the LMI type

For a Frame Relay network, the line protocol is the periodic exchange of local management interface (LMI) packets between the SmartNode and the Frame Relay provider equipment. If the SmartNode 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:

- **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.

```
node(cfg)#port serial 0 0
node(prt-ser)[0/0]#framerelay
node(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.

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

Enabling fragmentation

FRF.12 interface and end-to-end fragmentation of large IP packets is supported to reduce the delay imposed on voice packets on slow links (less than 512 kbps). As opposed to IP fragmentation, 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*(frm-rel)[***slot/port***]#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.
2	***node*(frm-rel)[***slot/port***]#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
3	***node*(frm-rel)[***slot/port***]#pvc** *dlci*	Enters the PVC configuration mode by assigning a DLCI number to be used on the specified virtual circuit.
4	***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 voice traffic is being given priority over data (command **source class local-voice priority**).

Note FRF.12 end-to-end fragmentation and FRF.12 interface fragmentation are incompatible. Thus make sure that both ends of a Frame Relay link run the same fragmentation mode.

Note When running data and voice over a Frame Relay link, it is advisable to only configure fragmentation for the PVC that carries data traffic. This way, fragmentation protocol overhead and fragmentation processing overhead is only spent for data traffic—voice packets (whose length should be smaller than the fragmentation length) do not consume processing power and protocol overhead for fragmentation.

The purpose of end-to-end FRF.12 fragmentation is to support real-time and non-real-time data packets on lower-speed links without causing excessive delay to the real-time data. The FRF.12 Implementation Agreement defines FRF.12 fragmentation. This standard was developed to allow long data frames to be fragmented into smaller pieces (fragments) and interleaved with real-time frames. In this way, real-time and non-real-time data frames can be carried together on lower-speed links without causing excessive delay to the real-time traffic.

End-to-end FRF.12 fragmentation is recommended for use on permanent virtual circuits (PVCs) that share links with other PVCs transporting voice and on PVCs transporting Voice over IP (VoIP).

The fragmentation size depends on the available bandwidth, the chosen codec, and its packet length:

- The less bandwidth available per call, the smaller the fragment size has to be configured.
- The shorter the voice packets, the smaller the fragment size can be configured.
- The smaller the fragment size, the bigger the overhead for long data packets.

The following table shows the minimum fragment size depending on the configured codec and its packet length without fragmenting the voice packets:

Codec (bytes)	Packet Period (ms)	Minimum Fragment Size
G.729	10	52
G.729	20	62
G.729	30	72
G.723	30	66
G.723	60	90
G.723	90	114
G.711	10	122
G.711	20	202
G.711	30	282

Entering Frame Relay PVC configuration mode

The permanent virtual circuit (PVC) is a virtual circuit that is permanently established. PVCs save bandwidth associated with circuit establishment and tear down in situations where certain virtual circuits must exist all the time.

The Frame Relay network provides a number of virtual circuits that form the basis for connections between stations attached to the same Frame Relay network.

The resulting set of interconnected devices forms a private Frame Relay group, which may be either fully interconnected with a complete mesh of virtual circuits, or only partially interconnected.

In either case, each virtual circuit is uniquely identified at each Frame Relay interface by a Data Link Connection Identifier (DLCI). In most circumstances, DLCIs have strictly local significance at each Frame Relay interface.

Assigning a DLCI to a specified Frame Relay sub interface 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.

```
node(cfg)#port serial 0 0
node(prt-ser)[0/0]#framerelay
node(frm-rel)[0/0]#pvc 1
node(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/PVC

Step	Command	Purpose
1	***node*(pvc)[*dlci*]#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.

```
node(cfg)#port serial 0 0
node(prt-ser)[0/0]#framerelay
node(frm-rel)[0/0]#pvc 1
node(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 28 below.

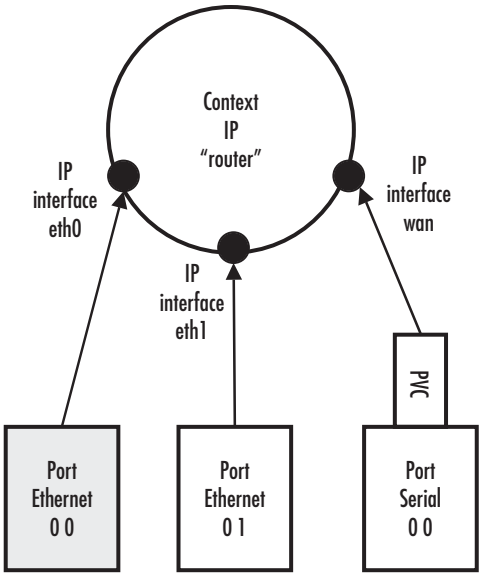


Figure 28. IP interface *wan* is bound to PVC 1 on port serial 0 0

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.

```
node(cfg)#port serial 0 0
node(prt-ser)[0/0]#framerelay
node(frm-rel)[0/0]#pvc 1
node(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.

```
node(cfg)#port serial 0 0
node(prt-ser)[0/0]#framerelay
node(frm-rel)[0/0]#pvc 1
node(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.

```
node(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.

```
node(cfg)#port serial 0 0
node(prt-ser)[0/0]#framerelay
node(frm-rel)[0/0]#pvc 1
```

```
node(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.

```
node(pvc)[1]#show running-config
Running configuration:
#-----------------------------------------------------------------#
#                                                                 #
# 2500                                                            #
…
pvc 1
  encapsulation rfc1490
  bind interface wan router
  shutdown
  exit
…
```

Debugging Frame Relay

A set of commands is available to check the status of the 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 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.

```
node>enable
node#configure
node(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
```

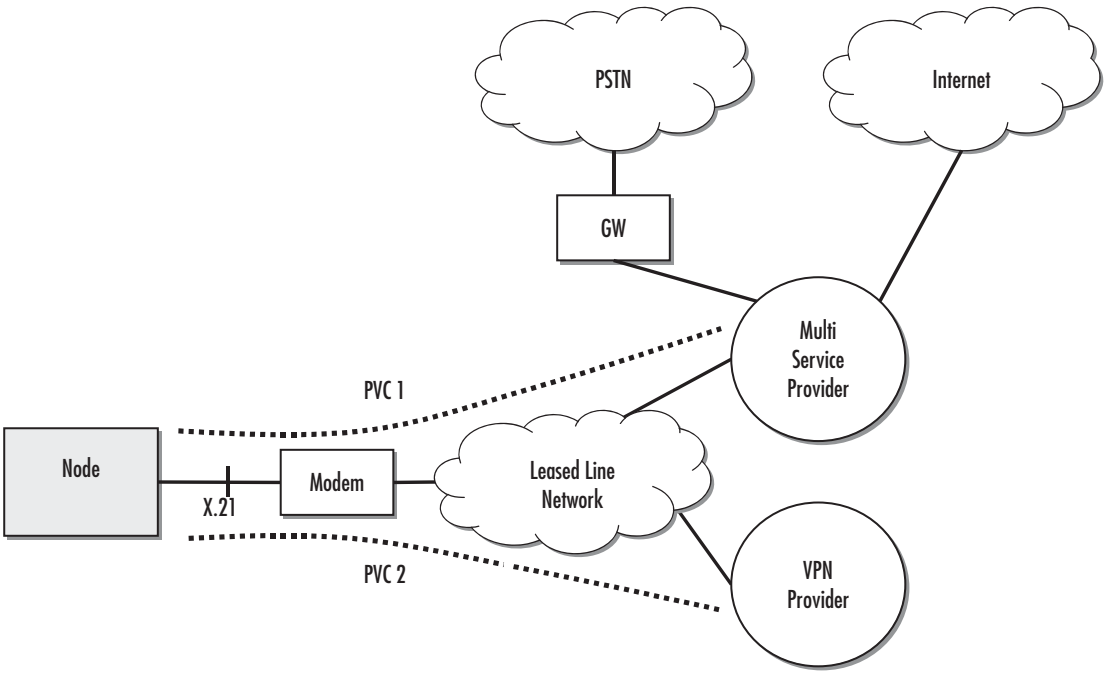


Figure 29. Typical Integrated Service Access Scenario with dedicated PVCs

Integrated service access

The example in figure 29 shows a typical integrated service access scenario, where different service providers are accessed via permanent virtual circuits (PVCs) on Frame Relay over the serial interface of a SmartNode.

The multi service provider (MSP) offers both Internet access and voice services based on IP. The virtual private network (VPN) provider offers secure interconnections of local access networks (LAN) via its public wide area network based on IP. Since both providers are working independently, the SmartNode needs a configuration, which has two dedicated PVCs on Frame Relay. The first PVC, labeled as PVC 1, connects to the MSP access device. The second PVC, labeled PVC 2, connects to the VPN provider access device on the leased line network.

A SmartNode is working as a DTE and accesses the leased line network via a leased line modem connected to the serial interface. The hardware port protocol X.21 is used on the serial interface on slot 0 and port 0.

Devices accessing the MSP and VPN services are attached to the 100 Mbps Ethernet port 0/0 on the SmartNode. For that reason, an IP context with three logical IP interfaces bound to Ethernet port 0/0, PVC 1 and PVC 2 on serial port 0/0 as shown in figure 29 has to be configured for the SmartNode. The IP interfaces are labeled to represent the function of their configuration. Hence Ethernet port 0/0 is named *lan*, PVC 1 is named *external* since external services are accessed via this PVC, and PVC 2 is named *internal* to indicate the private network interconnection via this PVC.

Between the leased line modem and the SmartNode, ANSI T.617 type of LMI packets have to be exchanged. In addition, the keep-alive interval has to be set to 20 seconds. To guarantee voice quality, fragmentation is enabled on the PVC which carries voice (PVC 1) and a service profile is assigned which gives priority to voices packets.

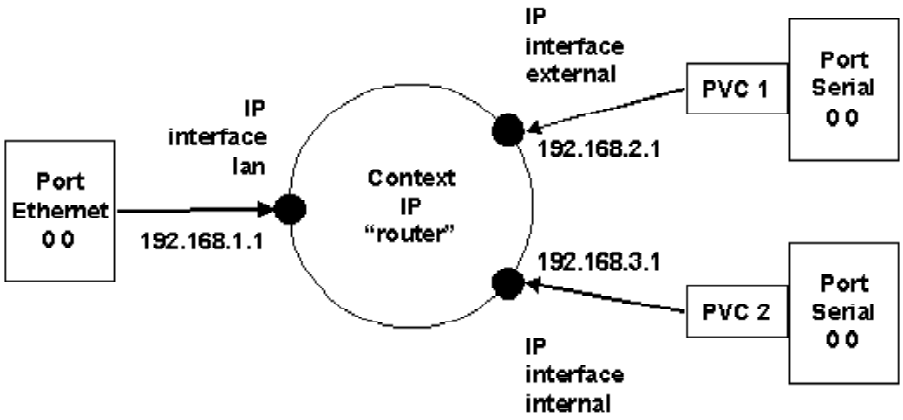


Figure 30. IP Context with logical IP interfaces bound to Ethernet port, serial port PVC 1 and PVC 2

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.

```
node>enable
node#configure
…
```

2. Set up the IP interface configuration first. Be aware that not all of the necessary settings are listed below.

```
node(cfg)#context ip router
node(ctx-ip)[router]#interface external
node(if-ip)[external]#interface internal
node(if-ip)[internal]#interface lan
node(if-ip)[lan]#exit
node(ctx-ip)[router]#interface internal
node(if-ip)[internal]#ipaddress 192.168.3.1 255.255.255.0
node(if-ip)[internal]#interface external
node(if-ip)[external]#ipaddress 192.168.2.1 255.255.255.0
node(if-ip)[external]#interface lan
node(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).

```
node(cfg)#profile service-policy VoicePrio
node(pf-srvpl)[VoicePr~]#rate-limit 512
node(pf-srvpl)[VoicePr~]#source class local-voice
node(src)[local-v~]#priority
node(src)[local-v~]#source class local-default
node(src)[local-d~]#priority
node(src)[local-d~]#source class default
…
```

4. Configure the serial interface settings.

```
node(cfg)#port serial 0 0
node(prt-ser)[0/0]#shutdown
node(prt-ser)[0/0]#encapsulation framerelay
node(prt-ser)[0/0]#hardware-port x21
node(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.

```
node(prt-ser)[0/0]#framerelay
node(frm-rel)[0/0]#lmi-type ansi
node(frm-rel)[0/0]#keepalive 20
node(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.

```
node(frm-rel)[0/0]#pvc 1
node(pvc)[1]#encapsulation rfc1490
node(pvc)[1]#fragment 300
node(pvc)[1]#bind interface external router
node(pvc)[1]#no shutdown
node(pvc)[1]#pvc 2
node(pvc)[2]#encapsulation rfc1490
node(pvc)[2]#bind interface internal router
node(pvc)[2]#no shutdown
…
```

7. Check that the Frame Relay settings are correct.

```
node(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
```

Example 1: Frame Relay on e1t1 without a channel-group

```
port e1t1 0 3
  port-type e1
  clock master
  framing crc4
  encapsulation hdlc
```

```
  hdlc
    encapsulation framerelay

    framerelay

      pvc 100
        encapsulation rfc1490
        bind interface pvc100 router
        no shutdown

port e1t1 0 0
  no shutdown
```

Example 2: Frame Relay on e1t1 with a channel-group

```
port e1t1 0 0
  port-type e1
  clock master
  framing crc4
  encapsulation channelized

  channel-group myGroup
    timeslots 13-17
    encapsulation hdlc

    hdlc
      encapsulation framerelay

      framerelay
        lmi-type gof
        keepalive 20

        pvc 100
          encapsulation rfc1490
          bind interface pvc100 router
          no shutdown

port e1t1 0 0
  no shutdown
```

Chapter 16 PRI port configuration

Chapter contents

Introduction 192
PRI port configuration task list 192
Enable/Disable PRI port 193
Configuring PRI port-type 193
Configuring PRI clock-mode 193
Configuring PRI line-code 193
Configuring PRI framing 194
Configuring PRI line-build-out (E1T1 in T1 mode only) 195
Configuring PRI used-connector (E1T1 in E1 mode only) 195
Configuring PRI application mode (E1T1 only) 195
Configuring PRI LOS threshold (E1T1 only) 196
Configuring PRI Loopback detection (E1T1 only) 196
Configuring PRI encapsulation 197
Create a Channel-Group 198
Configuring Channel-Group Timeslots 198
Configuring Channel-Group Encapsulation 198
Entering HDLC Configuration Mode 199
Configuring HDLC CRC-Type 199
Configuring HDLC Encapsulation 200
PRI Debugging 200
PRI Configuration Examples 201
Example 1: ISDN 202
Example 2: RBS without a channel-group 202
Example 3: RBS with a channel-group 202
Example 4: Frame Relay without a channel-group 203
Example 5: Framerelay with a channel-group 204
Example 6: PPP without a channel-group 204
Example 7: PPP with a channel-group 204

Introduction

This chapter provides an overview of the PRI (Primary Rate Interface) ports, their characteristics and the tasks involved in the configuration. The SmartNode devices know three different kinds of PRI ports, E1, T1 and E1T1 whereas an E1T1 port can either work as E1 or T1. This chapter describes the superset of all commands are available on the different PRI ports. If a command is only executable for a specific port then this circumstance will be noted or highlighted in the command description. Further will be explained here, how to prepare the ports for the usage of the different application protocols like ISDN, RBS, PPP or Frame Relay. Fore some applications there must be the possibility to access user defined sets of timeslots of an E1 or T1 port. On SmartNode's this feature is called a Channel Group and it will be described in this chapter as well.

Terminology

Hardware Type: Dependent on the device it can either be E1, T1 or E1T1. The Hardware Type and its belonging Slot and Port Number must be specified for entering the configuration mode of a port. It is not possible to change the Hardware Type, it is given by the system.

Port Type: This expression is used in relation with the E1T1 port and describes if the E1T1 port is currently running in E1 or in T1 mode. On an E1 or T1 port, the Port Type can not be changed, it is static and matches the Hardware Type.

PRI port configuration task list

This section describes the configuration tasks for the PRI port.

- Enable/Disable PRI port
- Configuring the PRI port type (E1T1 only)
- Configuring PRI clock mode
- Configuring PRI line code
- Configuring PRI framing (E1T1 only)
- Configuring PRI line build out (E1T1 in T1 mode only)
- Configuring PRI impedance/connector (E1T1 in E1 mode only)
- Configuring PRI application mode (E1T1 only)
- Configuring PRI LOS threshold (E1T1 only)
- Configuring PRI Loopback detection (E1T1 only)
- Configuring PRI encapsulation
- Create a Channel-Group
- Configuring channel-group timeslots
- Configuring channel-group encapsulation
- Entering HDLC configuration mode
- Configuration HDLC CRC-type
- Configuring HDLC encapsulation

- PRI Debugging

Enable/Disable PRI port

By default, the PRI port is disabled. The following command is used for enabling or disabling it.

Mode: port <hw-type> <slot> <port>

Step	Command	Purpose
1	[*name*] **(hw-type)**[slot/port]**# [no] shutdown**	Enable/Disable the PRI port. Default: *shutdown* (which is disabled)

Configuring PRI port-type

An E1T1 Port can either work in T1 or in E1 (G.704) mode. This mode can be changed dynamically as long as no encapsulation or encapsulation '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. Restriction: *Only available for e1t1 ports* Default: *e1*

Configuring PRI clock-mode

The PRI Port can either work in clock-master or in clock-slave mode. This setting defines the clock dependency of the internal data processing. In clock-master mode the internal data processing is running on an independent clock source. In clock-slave mode the clock source for internal data processing is recovered from the receive line interface. Be aware that always a port-pair of clock-master and clock-slave are connected together. In the other case the data transmission will fail due to bit failures. This command has also the option 'auto' that can be used if the application running on the port is also of an asymmetric nature like master/slave, server/client or user/net. Normally, the option 'auto' is used if the port is setup for ISDN. In this case, the clock mode will automatically derived from the Q.921 protocol. If the UNI-Side (User-Network Interface) of Q.921 is set to 'net', then clock mode of the port is automatically set to 'master' and if Q.921 is configured as 'user' it will be set to 'slave'.

Mode: port <hw-type> <slot> <port>

Step	Command	Purpose
1	[*name*] **(prt-e1t1)**[slot/port]**# clock {auto | master | slave}**	Configures the clock-mode of the port. Default: *master*

Configuring PRI line-code

Three different line codes can be selected on the PRI port whereas only 'ami' is standardized for E1 and T1. If the port is running in E1 mode, 'hdb3' is also configurable and in T1 mode 'b8zs'. If a linecode will be selected that is not standardized for the current port mode, an error message will be advised.

Mode: port <hw-type> <slot> <port>

Step	Command	Purpose
1	[*name*] **(prt-e1t1)**[slot/port]**# linecode {ami | b8zs | hdb3}**	Configures the line-code of the port. Default for e1: *hdb3* Default for t1: *b8zs*

Configuring PRI framing

Four framing formats are available for selection on the E1T1 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* and T1 has the framing option *esf* and *sf*.

- CRC4 (E1): Cyclic Redundancy Check 4. A CRC4 Multi-Frame consists of 16 continuous Basic-Frames. Each Multi-Frame can be divided into two Sub Multi-Frames. The first bit of Timeslot 0 of each even Sub Multi-Frame is called the C-Bit and belongs to the CRC4 check sum.
- ESF (T1): Extended Super Frame. The ESF if made up of 24 Basic-Frames. Each Basic-Frame includes one overhead bit, the F-Bit. The 24 F-Bits of one Extended Super Frame are used for synchronization (6 Bit), transmitting data link information (12 Bit) and for CRC6 calculation (6 Bit).
- SF (T1): Super Frame: The SF is made up of 12 Basic-Frames. Each Basic-Frame includes one overhead bit, the F-Bit. The 12 F-Bits of one Super-Frame represent the frame alignment pattern that is used for synchronization.
- Unframed: 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 | sf | unframed}**	Configures the framing of the port. Restriction: *Only available for e1t1 ports* E1 mode formats are: *crc4*, *non-*crc4, *unframed*. T1 mode formats are: *esf*, sf, *unframed*. Default for **e1**: *crc4* Default for **t1**: *esf*

Configuring PRI line-build-out (E1T1 in T1 mode only)

The line build out configuration is used in long haul applications to prevent cross talk in the far end device.

Mode: port 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. Restriction: *Only available for e1t1 ports in T1 mode.* Default for t1: *0* dB

Configuring PRI used-connector (E1T1 in E1 mode only)

If the E1T1 WAN-Card provides several line interface connector types this 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. Restriction: *Only available for e1t1 ports in E1 mode.* Default for e1: *rj45*

Configuring PRI application mode (E1T1 only)

The PRI port can be configured to work in either short-haul or in long-haul mode. **Short-haul** is the default application and should be used for transmission distances up to **180m/600ft**. For transmission distances up to **1800m/6000ft**, select the **long-haul** application.

Mode: port e1t1 <slot> <port>

Step	Command	Purpose
1	[*name*] **(prt-e1t1)**[*slot/port*]**#application {long-haul | short-haul}**	Specifies the e1/t1 application mode Restriction: *Only available for e1t1 ports* Default: *short-haul*

Configuring PRI LOS threshold (E1T1 only)

This command takes effect only if the PRI port is configured for **long-haul** applications. It specifies the sensitivity for **Loss Of Signal threshold**. A signal suffers more attenuation over long distances than over short distances. Therefore the LOS-Threshold must be set higher for longer transmission distances. This command has a default value of -46dB 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 Restriction: *Only available for e1t1 ports* Default: *-46dB*

Configuring PRI Loopback detection (E1T1 only)

In T1 mode the E1T1 PRI port has the capability for auto detection of inband sent loop back 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 Repetition Code
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 PRI 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. Restriction: *Only available for e1t1 ports* Default: disabled

Configuring PRI encapsulation

The PRI encapsulation command prepares the port for a specific application protocol. After the right encapsulation type has been set, the configuration mode command for the selected protocol can be executed for protocol specific configuration.

- channelized: This special encapsulation type pushes the port in mode where it is possible to setup an application for a user defined set of timeslots. Normally, all timeslots of a port are under full control of the application specified with the encapsulation command. In 'channelized' mode, an application uses only the specified timeslots. If the encapsulation is set to 'channelized', use the *channel-group* command to create a new Channel Group an to enter its configuration mode. In the Channel Group configuration mode, the same encapsulation types as on the port configuration mode are available again, except channelized.
- hdlc: Enables HDLC Framing on the selected port. After encapsulation hdlc has been specified, the hdlc configuration mode can be entered to configure hdlc specific parameters and to define the link layer protocol must run over hdlc.
- q921: This encapsulation type automatically binds the signaling timeslot (D-channel) of the selected port to the ISDN Layer 2 protocol. This is timeslot 16 for an E1 and timeslot 24 for a T1 port. If in the q921 configuration mode q931 is specified as next encapsulation, the control of all remaining timeslots (B-channels) is given to the ISDN Layer 3 protocol. For more information please see Chapter18, "ISDN Overview" on page 212 and Chapter*19,* "ISDN configuration" on page 217.
- rbs: Robbed Bit Signaling encapsulation is only available for T1 ports.
  On specifying this encapsulation type, all the 24 timeslots will be bound to the RBS protocol. Enter the RBS configuration mode for RBS specific configuration (see Chapter 20, "RBS configuration" on page 225).

Mode: port <hw-type> <slot> <port>

Step	Command	Purpose
1	**[***name***] (prt-e1t1)[**slot/port**]#[no] encapsulation {channelized | hdlc | q921 | rbs}**	Specifies the encapsulation type of the PRI port. Default: *no encapsulation*

Create a Channel-Group

If the desired encapsulated channel uses only selected time slots (not the entire PRI), then it is necessary to set up a channel-group. To create a channel-group, set the PRI port's encapsulation to *channelized*. (See section "Configuring PRI encapsulation".) 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

The encapsulation command prepares the Channel Group for a specific application protocol. After the right encapsulation type has been set, the configuration mode command for the selected protocol can be executed for protocol specific configuration.

- hdlc: Enables HDLC Framing on the selected Channel Group. After encapsulation hdlc has been specified, the hdlc configuration mode can be entered to configure hdlc specific parameters and to define the link layer protocol must run over hdlc. The number of selected timeslots in the Channel Group also defines the data transmission rate of the hdlc protocol (n * 64kBit/s).
- q921: This encapsulation type can only be chosen if on the Channel Group only one timeslot is selected. It is NOT possible to bind multiple timeslots to the q921 protocol.
- rbs: Robbed Bit Signaling encapsulation is only available for T1 ports.
  On specifying this encapsulation type, all the timeslots specified in the Channel Group will be bound to the

RBS protocol. Enter the RBS configuration mode for RBS specific configurations (see Chapter 20, "RBS configuration" on page 225).

Mode: channel-group *group-name*

Step	Command	Purpose
1	[name] **(ch-grp)**[group-name]**#[no] encapsulation {hdlc | q921 | rbs}**	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 e1t1" 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 e1t1" 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. For more details see Chapter 15, "Frame Relay configuration" on page 177 and Chapter *30,* "PPP configuration" on page 313.

Mode: hdlc

Step	Command	Purpose
1	[*name*] **(hdlc)#encapsulation {framerelay | ppp}**	Specifies the encapsulation type of hdlc. Default: *no encapsulation*

PRI Debugging

For the investigation of possible problems in link establishment, data transmission or synchronization, there exists a debug command with the options 'event' and 'error'. The command has a hierarchical characteristic and can be applied to all ports of given type on the whole device, or to all ports of slot or just to one specific port.

Mode: Operator execution

Step	Command	Purpose
1	[*name*]**#[no] debug *hw-type* [([<slot> | [<port>]]) | [[event] | [error]]]**	Enables/Disables the PRI event/error monitor for the device a slot or a port. Examples: **1) [no] debug e1t1** Enables/Disables the event and the error monitor for all e1t1 ports of the device. **2) [no] debug e1 event** Enables/Disables the event monitor for all e1 ports of the device. **3) [no] debug t1 error** Enables/Disables the error monitor for all t1 ports of the device. **4) [no] debug e1 3** Enables/Disables the event and error monitor for all e1 ports on slot 3. **5) [no] debug e1t1 1 event** Enables/Disables the event monitor for all e1t1 ports on slot 1. **6) [no] debug t1 2 error** Enables/Disables the error monitor for all t1 ports on slot 2. **7) [no] debug t1 0 0** Enables/Disables the event and error monitor for the t1 port 0 on slot 0. **8) [no] debug e1 1 0 event** Enables/Disables the event monitor for the e1 port 0 on slot 1. **9) [no] debug e1t1 2 0 error** Enables/Disables the error monitor for the e1t1 port 0 on slot 2.

Mode: Operator execution

Step	Command	Purpose
1	[*name*]**#show port hw-type [[<slot> <port>] | [detail <level>]]**	Prints information about the specified port with a given detail level.

PRI Configuration Examples

Here is a group of seven configuration examples.

- Example 1: ISDN
- Example 2: RBS without a channel-group
- Example 3: RBS with a channel-group

- Example 4: Frame Relay without a channel-group
- Example 5: Frame Relay with a channel-group
- Example 6: PPP without a channel-group
- Example 7: PPP with a channel-group

Example 1: ISDN

```
port e1t1 0 0
  port-type t1
  clock auto
  linecode b8zs
  framing esf
  encapsulation q921

  q921
    uni-side auto
    encapsulation q931

    q931
      protocol ni2
      uni-side net
      bchan-number-order ascending
      encapsulation cc-isdn
      bind interface pri00 switch

port e1t1 0 0
  no shutdown
```

Example 2: RBS without a channel-group

```
port e1t1 0 0
  port-type t1
  clock master
  linecode b8zs
  framing esf
  encapsulation rbs

  rbs
    protocol ground-start exchange
    encapsulation cc-rbs
    bind interface pri00 switch

port e1t1 0 0
  no shutdown
```

Example 3: RBS with a channel-group

```
port e1t1 0 0
  port-type t1
  clock master
  linecode b8zs
  framing esf
  encapsulation channelized

  channel-group group_1_8
    timeslots 1-8
    encapsulation rbs

    rbs
```

```
        protocol eam-wink-start
        encapsulation cc-rbs
        bind interface pri00_1_8 switch

    channel-group group_9_16
      timeslots 9-16
      encapsulation rbs

      rbs
        protocol ground-start exchange
        encapsulation cc-rbs
        bind interface pri00_9_16 switch

    channel-group group_17_24
      timeslots 17-24
      encapsulation rbs

      rbs
        protocol eam-double-wink-start
        encapsulation cc-rbs
        bind interface pri00_17_24 switch

  port e1t1 0 0
    no shutdown
```

Example 4: 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 5: 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 6: 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 7: 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 BRI port configuration

Chapter contents

Introduction 206
BRI port configuration task list 206
Enable/Disable BRI port 206
Configuring BRI clock-mode 206
Configuring BRI Power-Feed 207
Configuring BRI encapsulation 207
Creating a channel group 207
Configuring channel-group timeslots 208
Configuring channel-group encapsulation 208
Entering HDLC configuration mode 208
Configuring HDLC encapsulation 208
BRI Debugging 209
BRI Configuration Examples 210
Example 1: ISDN with auto clock/uni-side settings 210
Example 2: ISDN with manual clock/uni-side settings 210
Example 3: Multi-Link PPP over two B-Channels 211

Introduction

This chapter provides an overview of the BRI (Basic Rate Interface) ports, their characteristics and the tasks involved in the configuration. A BRI port supports two 64kbit/s B-channels for switched voice or data connections, one 16kbit/s D-channel for signaling and always-on data transfer. This results a usable data bit rate of 144kBit/s.

BRI port configuration task list

This section describes the configuration tasks for the BRI port.

- Enable/Disable BRI port
- Configuring BRI clock mode
- Configuring BRI Power-Feed
- Configuring BRI encapsulation
- BRI Debugging

Enable/Disable BRI port

By default, the BRI port is disabled. The following command is used for enabling or disabling it.

Mode: port bri <slot> <port>

Step	Command	Purpose
1	[*name*] **(prt-bri)**[slot/port]**# [no] shutdown**	Enable/Disable the selected port. Default: *shutdown* (which is disabled)

Configuring BRI clock-mode

The BRI Port can either work in clock-master or in clock-slave mode. This setting defines the clock dependency of the internal data processing. In clock-master mode the internal data processing is running on an independent clock source. In clock-slave mode the clock source for internal data processing is recovered from the receive line interface. Be aware that always a port-pair of clock-master and clock-slave are connected together. In the other case the data transmission will fail due to bit failures. This command has also the option 'auto' that can be used if the application running on the port is also of an asymmetric nature like master/slave, server/client or user/net. Normally, the option 'auto' is used if the port is setup for ISDN. In this case, the clock mode will automatically derived from the Q.921 protocol. If the UNI-Side (User-Network Interface) of Q.921 is set to 'net', then clock mode of the port is automatically set to 'master' and in the other case to 'slave'.

Mode: port bri <slot> <port>

Step	Command	Purpose
1	[*name*] **(prt-bri)**[slot/port]**# clock {auto | master | slave}**	Configures the clock-mode of the port. Default: auto

Configuring BRI Power-Feed

Enables the application of power on the BRI port to provide power to ISDN terminals. This command applies only if the port is clock master (network side). It is only available on products with an internal, configurable ISDN power supply.

Mode: port bri <slot> <port>

Step	Command	Purpose
1	[*name*] **(prt-bri)**[slot/port]**#[no] power-feed**	Enables/Disables power-feed on the selected port. Default: disabled

Configuring BRI encapsulation

The BRI encapsulation command prepares the port for a specific application protocol. After the right encapsulation type has been set, the configuration mode command for the selected protocol can be executed for protocol specific configuration.

- **q921:** This encapsulation type automatically binds the signaling timeslot of the selected port to the ISDN Layer 2 protocol. For the BRI port this is the 16kbit/s D-channel. If in the q921 configuration mode q931 is specified as next encapsulation, the control of the two remaining timeslots (B-channels) is given to the ISDN Layer 3 protocol. For more information please consult Chapter *19,* "ISDN configuration" on page 217.
- **channelized:** This special encapsulation type pushed the port in a mode where it is possible to setup an application for a user-defined timeslot. Normally, all timeslots of a port are under full control of the application specified with the encapsulation command. In 'channelized' mode, and application uses only the specified timeslot. If the encapsulation is set to 'channelized', use the channel-group command to create a new Channel Group and to enter its configuration mode. The requested can then be selected with the Channel Group's encapsulation command.

Mode: port bri <slot> <port>

Step	Command	Purpose
1	[*name*] **(prt-bri)**[slot/port]**#[no] encapsulation { channelized | q921}**	Specifies the encapsulation type of the BRI port. Default: *q921*

Creating a channel group

If the desired encapsulated channel uses only a selected time slot (not the entire BRI), then it is necessary to set up a channel-group. To create a channel-group, set the BRI port's encapsulation to **channelized**. (See s"Configuring BRI encapsulation".) 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 must be used.

Mode: port bri <slot> <port>

Step	Command	Purpose
1	[*name*] **(prt-bri)**[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 the timeslot for use in data transmission. On a BRI port, only the B-Channels can be selected (0 or 1).

Mode: channel-group <group-name>

Step	Command	Purpose
1	[*name*] **(ch-grp)**[*group-name*]**#[no] timeslots** *<timeslots>*	Selects the timeslot to be used. Default: no timeslots

Configuring channel-group encapsulation

The encapsulation command prepares the Channel Group for a specific application protocol. After the right encapsulation type has been set, the configuration mode command for the selected protocol can be executed for protocol specific configuration.

- **hdlc:** Enables HDLC Framing on the selected Channel Group. After encapsulation hdlc has been specified, the hdlc configuration mode can be entered to configure hdlc specific parameters and to define the link layer protocol must run over hdlc.

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 from the 'channel-group' configuration mode if the encapsulation is set to 'hdlc' and a timeslot has been specified. If the hdlc configuration mode is not accessible, it may be due to an invalid or incomplete configuration. In this case, an error message will be issued.

Mode: channel-group <group-name>

Step	Command	Purpose
1	[*name*] **(ch-grp)**[*group-name*]**#hdlc**	Enters the hdlc configuration mode.

Configuring HDLC encapsulation

The hdlc encapsulation command specifies what kinds of upper layer data are contained in the hdlc frames. Currently, only PPP can be chosen. For more details about PPP configuration, see Chapter 30, "PPP configuration" on page 313. Also, see "Example 3: Multi-Link PPP over two B-Channels" on page 211.

Mode: channel-group <group-name>

Step	Command	Purpose
1	[*name*] **(hdlc)#[no] encapsulation {ppp}**	Specifies the encapsulation type of hdlc. Default: no encapsulation

BRI Debugging

For the investigation of possible problems in link establishment, data transmission or synchronization, there exists a debug command with the options 'event' and 'error'. The command has a hierarchical characteristic and can be applied to all ports on the whole device, or to all ports of slot or just to one specific port. In addition, the 'show port' command can be used to printout information about the current configuration and about received and transmitted frames.

Mode: Operator execution

Step	Command	Purpose
1	[*name*]**#[no] debug *bri* [([<slot> | [<port>]]) | [[event] | [error]]]**	Enables/Disables the BRI event/error monitor for the device a slot or a port. Default: *no debug bri* Examples: **1) [no] debug bri** Enables/Disables the event and the error monitor for all bri ports of the device. **2) [no] debug bri event** Enables/Disables the event monitor for all bri ports of the device. **3) [no] debug bri error** Enables/Disables the error monitor for all bri ports of the device. **4) [no] debug bri 3** Enables/Disables the event and error monitor for all bri ports on slot 3. **5) [no] debug bri 1 event** Enables/Disables the event monitor for all bri ports on slot 1. **6) [no] debug bri 2 error** Enables/Disables the error monitor for all bri ports on slot 2. **7) [no] debug bri 0 0** Enables/Disables the event and error monitor for the bri port 0 on slot 0. **8) [no] debug bri 1 0 event** Enables/Disables the event monitor for the bri port 0 on slot 1. **9) [no] debug bri 2 0 error** Enables/Disables the error monitor for the bri port 0 on slot 2.

Mode: Operator execution

Step	Command	Purpose
1	[*name*]**#show port bri [[<slot> <port>] | [detail <level>]]**	Prints information about the specified port with a given detail level.

BRI Configuration Examples

- Example 1: ISDN with auto clock/uni-side settings
- Example 2: ISDN with manual clock/uni-side settings
- Example 3: Multi-Link PPP over two B-Channels

Example 1: ISDN with auto clock/uni-side settings

```
port bri 0 4
  power-feed
  encapsulation q921

  q921
    uni-side auto
    encapsulation q931

    q931
      protocol dss1
      uni-side net
      bchan-number-order ascending
      encapsulation cc-isdn
      bind interface bri04 switch

port bri 0 4
  no shutdown
```

Example 2: ISDN with manual clock/uni-side settings

```
port bri 0 4
  clock slave
  encapsulation q921

  q921
    uni-side user
    encapsulation q931

    q931
      protocol dss1
      uni-side user
      bchan-number-order ascending
      encapsulation cc-isdn
      bind interface bri04 switch

port bri 0 4
  no shutdown
```

Example 3: Multi-Link PPP over two B-Channels

```
context ip router
  interface wan-bri
  ipaddress 10.10.10.2 255.255.255.0
subscriber ppp bri_0_0
  dial in
  multilink max-links 2 fragmentation equal-distribution 320
  bind interface wan-bri router
port bri 0 0
  clock slave
  encapsulation channelized
  channel-group Group0
    timeslots 0
    encapsulation hdlc
    hdlc
      encapsulation ppp
      bind subscriber bri_0_0
  channel-group Group1
    timeslots 1
    encapsulation hdlc
    hdlc
      encapsulation ppp
      bind subscriber bri_0_0
port bri 0 0
  no shutdown
```

Chapter 18 ISDN Overview

Chapter contents

Introduction 213
ISDN reference points 213
Possible SmartNode port configurations 214
ISDN UNI Signaling 214
ISDN Configuration Concept 216
ISDN Layering 216

Introduction

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

ISDN ports are the physical ISDN connections on the SmartNode devices. There are two types of ISDN ports:

- The ISDN basic rate interface (BRI), and
- The ISDN primary rate interface (PRI).

A BRI port supports two 64kbit/s B-channels for switched voice or data connections, one 16kbit/s D-channel for signaling and always-on data transfer. BRI ports are sometimes called S0 ports. The related PSTN access service is also called Basic Rate Access (BRA).

The PRI port supports thirty 64kbit/s B-channels, one 64kbit/s D-channel and one synchronization timeslot on a standard E1 (G.704) physical layer. PRI ports are also called S2m ports. The related PSTN access service is also called Primary Rate Access (PRA).

ISDN reference points

The ISDN standards define a number of reference points on the interfaces between the various equipment types on an ISDN access line. Figure 31 illustrates these reference points. The understanding of these reference points and where they are located is necessary for the configuration of the SmartNode ISDN ports.

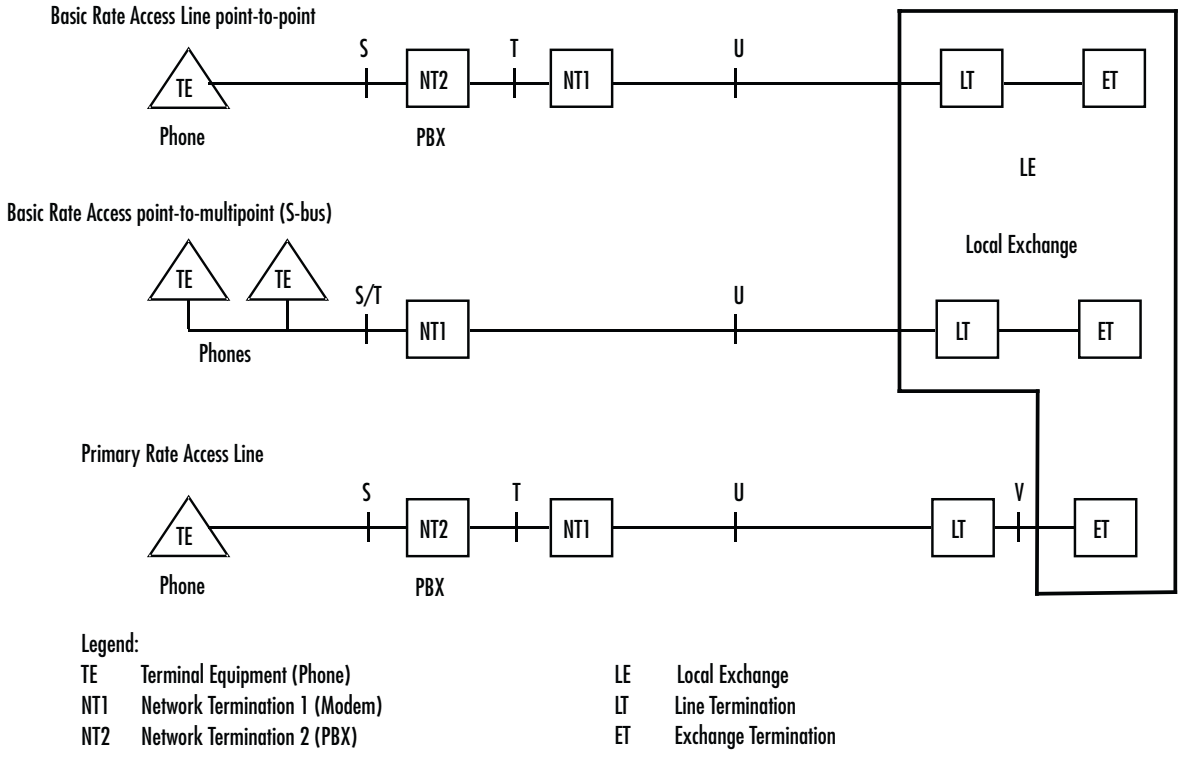


Figure 31. ISDN reference points

The S reference point is on the subscriber interface. This is the typical 4-wire connection between an ISDN phone and an ISDN PBX. Be aware that many ISDN PBX vendors use non-standard proprietary 2-wire interfaces to connect the Terminals to the PBX.

The T reference point is on the trunk interface of a PBX. This is the standard 4-wire interface between the PBX and the network termination unit (NTU) also known as NT1 in standard terminology. The ISDN layer 2 protocol at this point is in point-to-point mode between the NTU and the PBX.

The 4-wire layer 1 specification S and T interfaces is foreseen for in-house installations and carries a maximum of 150 meters.

The S/T reference point is on a point-to-multipoint S-Bus. Here several terminals are connected directly to the same BRI NTU. The S and T reference points are “collapsed”. The NT2 is not represented by any equipment unit.

The U reference point is on the transmission side of the NTU designed to carry the ISDN line over the last mile. For basic rate interfaces this is typically a DSL technology working on legacy copper pairs over a distance up to 12 kilometers. For primary rate lines, DSL, coax and fiber transmission is in use. In most European countries the U interface is not accessible to the subscriber, the operator always provides the NT1. In the US and some other countries the NT1 can be integrated into the NT2, i.e. the PBX is connected directly to the U interface.

The V reference point is typically a y-wire interface between the line card of the public switch and the 2 Mbps transmission equipment which transports the PRI signal over copper (DSL), coax or fiber.

Possible SmartNode port configurations

The SmartNode ISDN ports can be configured for connection to S, T, S/T, and V interfaces. Refer to figure 33, which illustrates some of the possible network integration options.

ISDN UNI Signaling

ISDN is a User-Network Interface (UNI) signaling protocol with a user and a network side. The user side is implemented in ISDN terminals (phones, terminal adapters, etc.) while the network side is implemented in the exchange switches of the network operator. Both sides have different signaling states and messages. SmartWare ISDN ports can be configured to work as user (USR) or network (NET) interfaces.

A SmartNode in some applications does not replace a standard ISDN equipment (PBX or Terminal) but is inserted between an existing NT and PBX. In such cases the SmartNode ISDN ports are configured to operate the opposite side of the connected equipment as illustrated in figure 33.

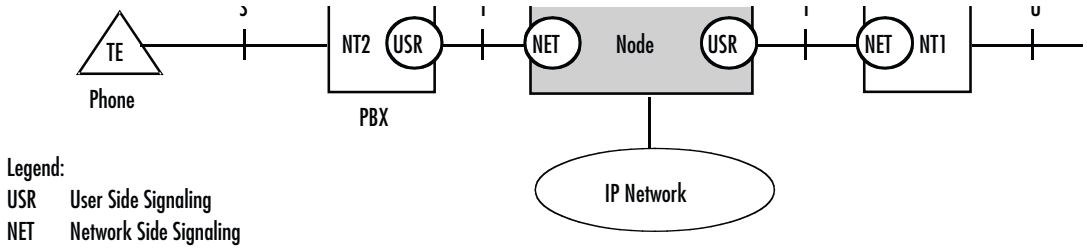


Figure 32. ISDN signaling side

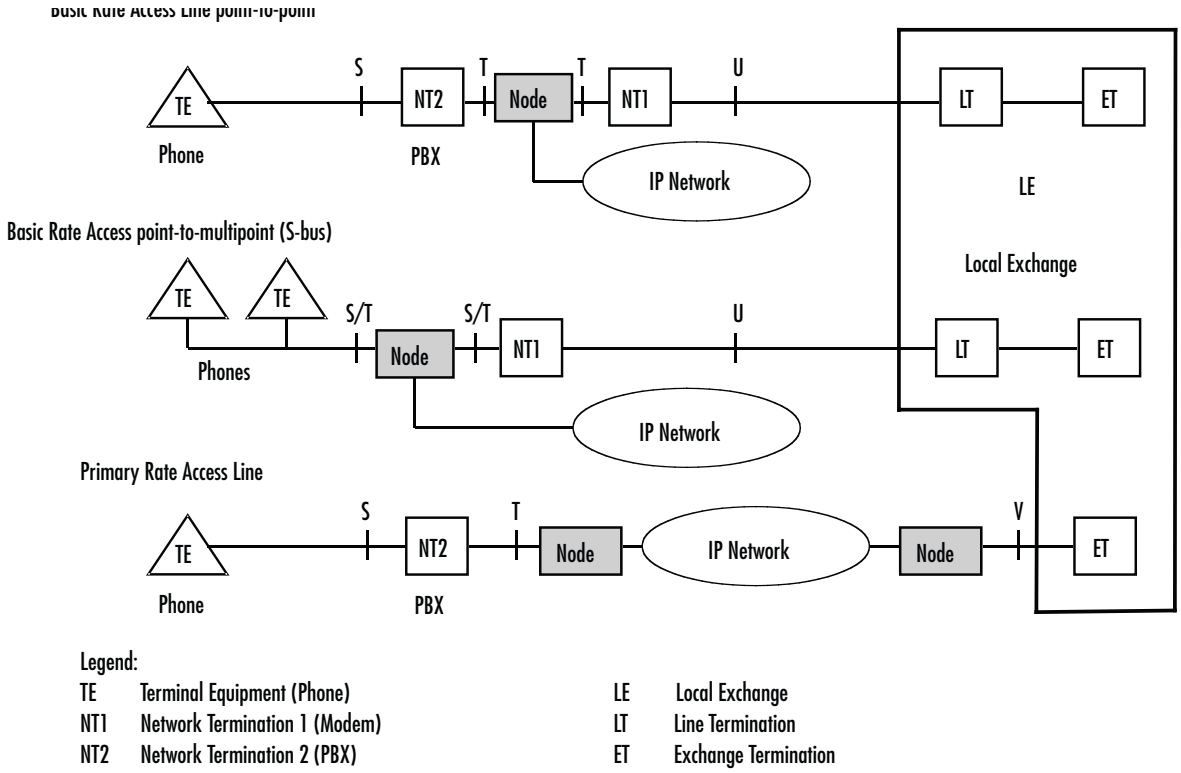


Figure 33. Integration of ISDN access lines

IMPORTANT

Port activation deactivation—ISDN ports can be configured while they are active. However they will be internally disabled to modify the configuration and then re-enabled. All active calls on the port are dropped during this process. Configuration changes should only be performed during planned down times.

IMPORTANT

Reference clock source and synchronization—The SmartNode uses a single reference clock source for the synchronization of the 64kbit/s PCM channels on the ISDN ports and in the CS context. This reference clock source can be internal or it can be derived from one of the ISDN ports. If the clock reference is not configured in accordance with the network environment, clock slips and related voice quality degradations can occur. Refer to chapter 31, "CS context overview" on page 339 on how to configure the reference clock

IMPORTANT

Connector pin-out and short circuits—Some of the SmartNode ISDN BRI ports are configurable to operate as network or terminal ports. The pin-out of the sockets is switched according to this configuration. Wrong port configurations, wrong cabling or wrong connections to neighboring equipment can lead to short circuits in the BRI line powering. Refer to the HW installation guide and the port configuration sections below to avoid misconfigurations.

ISDN Configuration Concept

ISDN Layering

ISDN consists of 3 layers. Each layer has its own parameters that need to be configured.

- Layer 1, often called the physical layer, is responsible to transport single bits between two systems. Layer 1 does not guarantee that a message can be transmitted without errors.

  Parameters: Clock mode, line codes.

- Layer 2 allows a station to reliably send messages to another station using the D channel. Layer 2 implements flow control, error detection and correction (retransmission) as well as addressing mechanism to direct messages to individual devices.

  Parameters: point-to-point or point-to-multipoint mode, network/user side, permanent layer 2 enabled.

- Layer 3 does send and receive application level messages (i.e. call control). It cares for sending broadcast messages and collecting the individual results of the attached devices. It also handles the assignment of the B channels.

  Parameters: network/user side, protocol (i.e. DSS1), maximum number of channels.

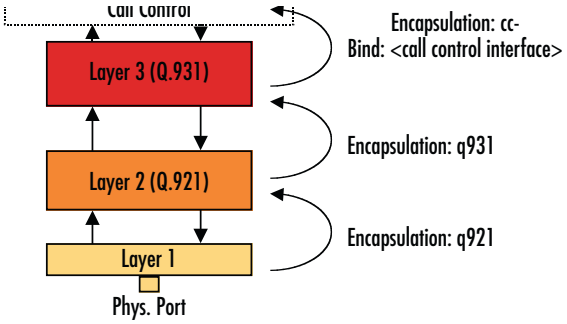


Figure 34. ISDN layering model

The layered model of ISDN is reflected in the configuration by the use of different modes for each layer. The layers are connected by using encapsulations and bindings. The encapsulation defines what the next higher layer protocol will be. On the topmost layer, the binding finally selects a logical interface to connect the port to. For more information how to configure and setup the physical ports for ISDN, please see Chapter 17, "BRI port configuration" on page 205 and Chapter 16, "PRI port configuration" on page 191. Detailed information about Q.921 and Q.931 configuration are available in Chapter 19, "ISDN configuration" on page 217.

Chapter 19 ISDN configuration

Chapter contents

Introduction 218
ISDN configuration task list 218
- Enter Q.921 configuration mode 218
- Configuring Q.921 parameters 218
- Configuring Q.921 encapsulation 219
- Enter Q.931 configuration mode 219
- Configuring Q.931 parameters 220
- Configuring Q.931 encapsulation 222
- Debugging ISDN 222
- ISDN Configuration Examples 223

Introduction

This chapter describes the configuration of the Q.921 and Q.931 protocol and how to bind the ISDN protocol to an application like the Call Control. To get an overview of the ISDN protocol and the layered configuration model of SmartWare, please see Chapter 18, "ISDN Overview" on page 212. In this chapter it is supposed, the lower layer on which ISDN will be setup is correctly configured. If ISDN has to run on a TDM port like BRI or PRI, please see Chapter 17, "BRI port configuration" on page 205 or Chapter 16, "PRI port configuration" on page 191.

ISDN configuration task list

Configuring ISDN typically consists of the following tasks:

- Enter Q.921 configuration mode
- Configuring Q.921 parameters
- Configuring Q.921 encapsulation
- Enter Q.931 configuration mode
- Configuring Q.931 parameters
- Configuring Q.931 encapsulation

Enter Q.921 configuration mode

Normally, Q.921 is running as ISDN Layer 2 protocol on a BRI or PRI port. But it is also possible another protocol is using Q.921 as its next encapsulation step an then Q.921 will not be configured out of a port context. That means, Q.921 encapsulation can be configured in different configuration modes. For this reason, the command description below refers to the configuration mode in which Q.921 can be enabled by setting the encapsulation to 'q921'. This configuration mode is called here 'base-mode' but it is only an alias for the real mode. Once encapsulation q921 has been configured, the Q.921 configuration mode can be entered.

Mode: base-mode

Step	Command	Purpose
1	**node(base-mode)]#[no] encapsulation q921**	Enables/Disables Q.921
2	**node(base-mode)]#q921**	Enter the Q.921 configuration mode

Configuring Q.921 parameters

This chapter provides an overview of the Q.921 configuration parameters, their syntax and possible restrictions. In case of ISDN, Q.921 settings apply to both BRI and PRI ports. They are defined in the q921 mode. To use Q921, the lower layer encapsulation must be set to q921.

Mode: q921

Step	Command	Purpose
1	***node*(q921)[slot/port]#protocol pp** or ***node*(q921)[slot/port]#protocol pmp**	Specify Q.921 operating mode (Default: BRI: pmp, PRI: pp). The Q.921 protocol running on BRI ports can operate in point-to-point (pp) or point-to-multi-point (pmp) mode. Point-to-multipoint is used to connect multiple terminals to an ISDN S-Bus. In some cases small PBXs are also connected to the public ISDN in point-to-multi-point mode. Point-to-point is typically used to connect PBXs to a public or private ISDN. The Q.921 protocol of PRI ports always run in point-to-point (pp) mode.
2	***node*(q921)[slot/port]#uni-side auto** or ***node*(q921)[slot/port]#uni-side net** or ***node*(q921)[slot/port]#uni-side user**	Specify the UNI side of the interface (Default: auto) If layer1 clock mode is not defined or set to auto this setting also specifies the clock mode for layer1. NET: clock mode = master USR: clock mode = slave If set to auto the UNI side setting is taken from layer3.
3	***node*(q921)[slot/port]#[no] permanent-layer2**	Enables the Q.921 permanent activity (Default: disabled). By default, the Q.921 protocol is not enabled permanently, i.e. the first call enables it.

Configuring Q.921 encapsulation

This command specifies the next protocol or application has to be attached to the Q.921 protocol. In case of ISDN this will always be the Q.931 protocol but in a distributed system for example, it could also be a network protocol.

Mode: q921

Step	Command	Purpose
1	***node*(q921)[*slot/port*]#[no] encapsulation q931**	Enables/Disables the next application or protocol. Currently only Q.931 is supported.

Enter Q.931 configuration mode

Normally, Q.931 is running as ISDN Layer 3 protocol on Q.921. But it is also possible another protocol is using Q.931 as its next encapsulation step an then Q.931 will not be configured out of the Q.921 context. That means, Q.931 encapsulation can be configured in different configuration modes. For this reason, the

command description below refers to the configuration mode in which Q.931 can be enabled by setting the encapsulation to 'q931'. This configuration mode is called here 'base-mode' but it is only an alias for the real mode. Once encapsulation q931 has been configured, the Q.931 configuration mode can be entered.

Mode: base-mode

Step	Command	Purpose
1	[*name*](base-mode)]#[no] encapsulation q931	Enables/Disables Q.931
2	[*name*](base-mode)]#q931	Enter the Q.931 configuration mode

Configuring Q.931 parameters

This chapter provides an overview of the Q.931 configuration parameters, their syntax and possible restrictions. In case of ISDN, Q.931 settings apply to both BRI and PRI ports. They are defined in the q931 mode. To use Q931, the lower layer encapsulation must be set to q931.

Note QSIG is an ISDN based protocol for signaling between nodes of a Private Integrated Services Network. The formal name of the signaling system by ISO / IEC is PSS1. Both names will co-exist and QSIG will continue to be used as the marketing name.

Mode: q931

Step	Command	Purpose
1	***node*(q931)[slot/port]#protocol dss1** or ***node*(q931)[slot/port]#protocol pss1** or ***node*(q931)[slot/port]#protocol ni2** or ***node*(q931)[slot/port]#protocol ntt** or ***node*(q931)[slot/port]#protocol dms-100**	Specify the ISDN layer 3 protocol (Default: BRI: dss1, E1: dss1, T1: ni2) The ISDN layer 3 is the network signaling protocol. SmartWare ISDN supports: • Euro-ISDN (E-DSS1) • Q.SIG (PSS1) • National ISDN (NI2) • Nippon Telecom NTT for BRI • Nortel Dms-100 for T1 The layer 3 signaling must correspond to the connected ISDN equipment or network.

Step	Command	Purpose
2	***node*(q931)[slot/port]#signalling-rule etsi** or ***node*(q931)[slot/port]#signalling-rule pss1old** or ***node*(q931)[slot/port]#no signalling-rule**	Specify channel numbering (Default: etsi) Some older Q-SIG variants make use of a channel numbering scheme that differs from the standard ETSI method. In most cases the ETSI numbering applies. Unless the connected ISDN devices and configured protocols require a different scheme, make sure the numbering is set to ETSI.
3	***node*(q931)[slot/port]#uni-side net** or ***node*(q931)[slot/port]#uni-side user**	Specify the UNI side of the interface. If not defined on layer2 (q921 mode) this setting also specifies the UNI side setting for layer2. The layer 2 settings also apply to Q.SIG (PSS1) interfaces. Make sure that the device connected to a SmartNode ISDN port is operating the opposite side of the configured uni-side.
4	***node*(q931)[slot/port]#max-calls** *number-of-calls* or ***node*(q931)[slot/port]#no max-calls**	Limits the total number of concurrent calls on the port. The **no** form of the command restores the default settings. **Note** if the channel-range and max-calls command are used simultaneously, the lower number of channels is the limiting parameter.
5	***node*(prt-pstn)[*slot/port*]#channel-range** *min max* *or* ***node*(prt-pstn)[*slot/port*]#no channel-range**	Specify B-channel range to be used on a PRI port (Default: E1: 0-31, T1: 0-23) Limits the time-slots to be used for calls to the range between *min* and *max*. This is in some cases required for interoperability with ISDN services that impose the same limitations. Call slots outside the defined range are rejected (busy line). If no range is defined (Default) all 30 (T1: 23) time-slots are available for use. The **no** form of the command restores the default settings.

Step	Command	Purpose
6	***node*(q931)[slot/port]# bchan-number-order ascending** or ***node*(q931)[slot/port]#bchan-number-order ascending-cyclic** or ***node*(q931)[slot/port]#bchan-number-order descending** or ***node*(q931)[slot/port]#bchan-number-order descending-cyclic**	Specify B-channel allocation strategy (Default: ascending) The numbering mode defines how the available time slots are filled. The cyclic modes use a "round-robin" implementation. The "up" and "down" modes define whether the time slots are filled at the lowest or highest available slot, i.e. *up* means that always the lowest available slot is used, *down* uses always the highest available slot.

Configuring Q.931 encapsulation

This command specifies the next protocol or application has to be attached to the Q.931 protocol. In case of ISDN this will always be the CC-ISDN (Call Control) application. For this case also a binding to a pre-created ISDN interface is necessary. For information about creation and configuration of an ISDN interface please see Chapter 34, "ISDN interface configuration" on page 390.

Mode: q931

Step	Command	Purpose
1	***node*(q931)[*slot/port*]#[no] encapsulation cc-isdn**	Enables/Disables the next application or protocol. Currently only CC-ISDN is supported.
2	***node*(q931)[*slot/port*]#[no] bind interface** *if-name*	Bind the Q.931 protocol to an existing call control interface.

Debugging ISDN

For the investigation of possible Q.921/Q.931 protocol problems or to get call signaling information, there exists a debug command with the options 'event' and 'error'. The command can be applied to the port on which ISDN is configured and has a further option to switch on or off a specific ISDN layer. In addition, the 'show port isdn' command can be used to printout information about the current state and statistic information about received and transmitted frames.

Mode: Operator execution

Step	Command	Purpose
1	***node*#debug isdn {event | error}** *slot port* **{all | layer2 | layer3}**	Enables/Disables the ISDN event/error monitor

Mode: Operator execution

Step	Command	Purpose
1	***node*#show port isdn** [*slot port*] [**detail** <-*level*>]	Show the status of one or more ISDN ports. If the optional arguments slot/port are omitted the status of all ISDN ports is displayed. *Level* could be 1 to 5. Level 1 shows less, level 5 shows all available information. Default level is 3.

ISDN Configuration Examples

Example: Configuring BRI port as Euro-ISDN interface

The following example shows how to configure port 0/0 as a Euro ISDN interface with user side signaling.

```
172.16.40.71(cfg)#port bri 0 0
172.16.40.71(prt-bri)[0/0]#q921
172.16.40.71(q921)[0/0]#q931
172.16.40.71(q931)[0/0]#uni-side user
172.16.40.71(q931)[0/0]#encapsulation cc-isdn
172.16.40.71(q931)[0/0]#bind interface bri00
172.16.40.71(q931)[0/0]#exit
172.16.40.71(q921)[0/0]#exit
172.16.40.71(prt-bri)[0/0]#no shutdown
```

Example: being clock slave on uni network interface

The following example shows how to configure both ports of a SmartNode with network signaling but receive the clock (via port 0) from the peer. The peer must be configured accordingly, i.e. port 0 as USR/clock master and port 1 NET/clock slave.

```
172.16.40.71(cfg)#port bri 0 0
172.16.40.71(prt-bri)[0/0]#clock slave
172.16.40.71(prt-bri)[0/0]#q921
172.16.40.71(q921)[0/0]#q931
172.16.40.71(q931)[0/0]#uni-side net
172.16.40.71(q931)[0/0]#encapsulation cc-isdn
172.16.40.71(q931)[0/0]#bind interface bri00
172.16.40.71(q931)[0/0]#exit
172.16.40.71(q921)[0/0]#exit
172.16.40.71(prt-bri)[0/0]#no shutdown

172.16.40.71(cfg)#port bri 0 1
172.16.40.71(prt-bri)[0/0]#q921
172.16.40.71(q921)[0/0]#q931
172.16.40.71(q931)[0/0]#uni-side net
172.16.40.71(q931)[0/0]#encapsulation cc-isdn
172.16.40.71(q931)[0/0]#bind interface bri01
172.16.40.71(q931)[0/0]#exit
172.16.40.71(q921)[0/0]#exit
172.16.40.71(prt-bri)[0/0]#no shutdown
```

Example: QSIG

Assume the scenario as illustrated in figure 35:

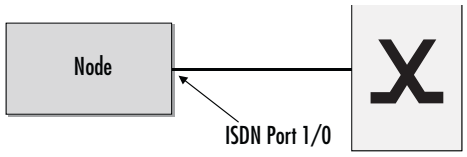


Figure 35. PBX connected to ISDN port 1/0

Configure the ISDN port 1/0 to work as a Q-SIG master port but clock-slave and allow a maximum of eight parallel B-channel connections.

```
172.16.40.71(cfg)#port e1 1 0
172.16.40.71(prt-e1)[1/0]#clock slave
172.16.40.71(prt-e1)[1/0]#q921
172.16.40.71(q921)[1/0]#q931
172.16.40.71(q931)[1/0]#uni-side net
172.16.40.71(q931)[1/0]#protocol pss1
172.16.40.71(q931)[1/0]#signalling-rule etsi
172.16.40.71(q931)[1/0]#max-channels 8
172.16.40.71(q931)[0/0]#exit
172.16.40.71(q921)[0/0]#exit
172.16.40.71(prt-e1)[0/0]#no shutdown
```

Example: PRI

Configure PRI port 1/0 as clock master. From the Local Exchange timeslots 1 through 20 are available and the total number of concurrent calls shall be limited to 10. Use down-cyclic channel numbering.

```
172.16.40.71(cfg)#port e1 1 0
172.16.40.71(prt-e1)[1/0]#q921
172.16.40.71(q921)[1/0]#q931
172.16.40.71(q931)[1/0]#uni-side net
172.16.40.71(q931)[1/0]#max-channels 10
172.16.40.71(q931)[1/0]#channel-range 1 20
172.16.40.71(q931)[1/0]#bchan-number-order descending-cyclic
172.16.40.71(q931)[0/0]#exit
172.16.40.71(q921)[0/0]#exit
172.16.40.71(prt-e1)[0/0]#no shutdown
```

Chapter 20 RBS configuration

Chapter contents

Introduction 226
RBS configuration task list 226
Enter RBS configuration mode 226
Configuring RBS protocol 226
Configuring RBS encapsulation 227
Debugging RBS 227
RBS Configuration Examples 228

Introduction

This chapter describes the configuration of the Robbed Bit Signaling (RBS) protocol and how to bind it to the Call Control application. RBS is used on T1 links to provide per-channel circuit signaling information. In this application no common signaling channel is used like in ISDN, each channel (Time Slot) is carrying its signaling information by itself. For this purpose, in every sixth frame the least significant bit of each timeslot is used (robbed) to transmit the signaling state. In the Super Frame (SF) format that is built on 12 basic frames, the bit robbed from the 6th frame is called the A-Bit and the bit robbed from the 12th frame is called the B-Bit. If the Extended Super Frame (ESF) format issued, that exists on 24 basic frames, the robbed bits are called A-Bit (6th frame), B-Bit (12th frame), C-Bit (18th frame) and D-Bit (24th frame). The information carried in these 2/4 bits is representing the current signaling state in a format it is known from the Analog Telephony (FXS/FXO). These states are for example On-Hook, Off-Hook and Ringing.

RBS configuration task list

Configuring RBS typically consists of the following tasks:

- Enter RBS configuration mode
- Configuring RBS protocol
- Configuring RBS encapsulation

Enter RBS configuration mode

There are two different ways how to use RBS. First, RBS encapsulation can be directly configured on the requested T1 port. In this case, all timeslots will use the same configured RBS protocol and will be bound to the same Call Control interface. But if not all timeslots of a T1 port have to be configured for RBS or some timeslots have to use a different RBS protocol or different groups of timeslots have to be bound to different Call Control interfaces, then the channelized port configuration model must be selected. For more information about the channelized model and the creation of channel groups, please consult Chapter16, "PRI port configuration" on page 191. Because RBS encapsulation can be set in different configuration modes (Port and Channel Group), an independent mode name 'base-mode' is used in the command description below. It refers to the real mode where encapsulation 'rbs' can be configured.

Mode: base-mode

Step	Command	Purpose
1	**node(base-mode)]#[no] encapsulation rbs**	Enables/Disables RBS
2	**node(base-mode)]#rbs**	Enter the RBS configuration mode

Configuring RBS protocol

RBS knows several different signaling protocols. Dependent on the application requirements the right one must be selected.

- Loop Start: It is the most common protocol and primarily used for local loop services. The protocol is asymmetric what means, the exchange and the subscriber side are different. Always an Exchange/Subscriber pair must be connected together. There is a provisioning for ring indication in this protocol.

- Ground Start: This protocol is commonly used for local loop PBX services. The protocol is asymmetric what means, the exchange and the subscriber side are different. Always an Exchange/Subscriber pair must be connected together. There is a provisioning for ring indication in this protocol.
- E&M Wink Start: This protocol is used between exchanges, is symmetric and has NO provisioning for ring indication. The 'wink' serves as an indication that the terminating side is ready to receive the called party number, it is analogous to the dial tone.
- E&M Immediate Start: This protocol is almost the same as E&M Wink Start but the originating side must have the capability for inband dial tone detection due to a missing 'ready to receive digits' indication.
- E&M Double Wink Start: This protocol is almost the same as E&M Wink Start with the difference after the terminating side has received all digits of the called party number, it sends back an acknowledge 'wink' to the originating side.

Mode: rbs

Step	Command	Purpose
1	**node(rbs)]#[no] protocol {loop-start {exchange | subscriber} | ground-start {exchange | subscriber} | eam-double-wink-start | eam-immediate-start | eam-wink-start}**	Selects the RBS protocol.

Configuring RBS encapsulation

This command specifies the next protocol or application has to be attached to the RBS protocol. Here it will always be the CC-RBS (Call Control) application and also a binding to a pre-created RBS interface is necessary. For information about creation and configuration of a RBS interface please consult Chapter 37, "RBS interface configuration" on page 426.

Mode: rbs

Step	Command	Purpose
1	**node(rbs)]#[no] encapsulation cc-rbs**	Enables/Disables the next application or protocol. Only CC-RBS is supported.
2	**node(rbs)]#[no] bind interface** *interface*	Bind the RBS protocol to an existing call control interface.

Debugging RBS

For the investigation of possible RBS protocol problems or to get information about the call signaling state, there exist two debug commands with the options 'event' and 'error'. The first command is called 'debug cas' (Channel Associated Signaling) and outputs information about sent and received A, B, C and D bits as well as information about the debouncing state. Debouncing of the received signaling state bits is necessary due to possible transmission failures on the TDM line. The second debug command is called 'debug rbs' and outputs information about call signaling state changes (On-Hook, Off-Hook, Ringing, Wink).

Mode: Operator execution

Step	Command	Purpose
1	***node*#[no] debug cas {event | error}**	Enables/Disables CAS event/error monitor

Mode: Operator execution

Step	Command	Purpose
1	***node*#[no] debug rbs {event | error}**	Enables/Disables RBS event/error monitor

RBS Configuration Examples

Example: Configuring RBS Ground Start on a E1T1 port

```
port e1t1 0 0
  port-type t1
  clock slave
  linecode b8zs
  framing esf
  encapsulation rbs

  rbs
    protocol ground-start subscriber
    encapsulation cc-rbs
    bind interface RBS00 switch

port e1t1 0 0
  no shutdown
```

Example: Configuring different RBS protocols with a Channel Group on an E1T1 port

```
port e1t1 0 0
  port-type t1
  clock slave
  linecode b8zs
  framing esf
  encapsulation channelized

  channel-group RBS_GROUP_1_8
    timeslots 1-8
    encapsulation rbs

    rbs
      protocol eam-wink-start
      encapsulation cc-rbs
      bind interface RBS00_1_8 switch

  channel-group RBS_GROUP_9_16
    timeslots 9-16
    encapsulation rbs

    rbs
      protocol eam-immediate-start
      encapsulation cc-rbs
      bind interface RBS00_9_16 switch
```

```
  channel-group RBS_GROUP_17_24
    timeslots 17-24
    encapsulation rbs

    rbs
      protocol eam-double-wink-start
      encapsulation cc-rbs
      bind interface RBS00_17_24 switch

port e1t1 0 0
  no shutdown
```

Chapter 21 DSL Port Configuration

Chapter contents

Introduction 231
Line Setup 231
Configuring PPPoE 231
Configuration Summary 232
Setting up permanent virtual circuits (PVC) 233
- Using PVC channels in bridged Ethernet mode 233
- Using PVC channels with PPPoE 233
- Diagnostics 234

Troubleshooting DSL Connections 234

Introduction

This chapter provides an overview of the DSL ports (ADSL and G.SHDSL), their characteristics and the tasks involved in the configuration.

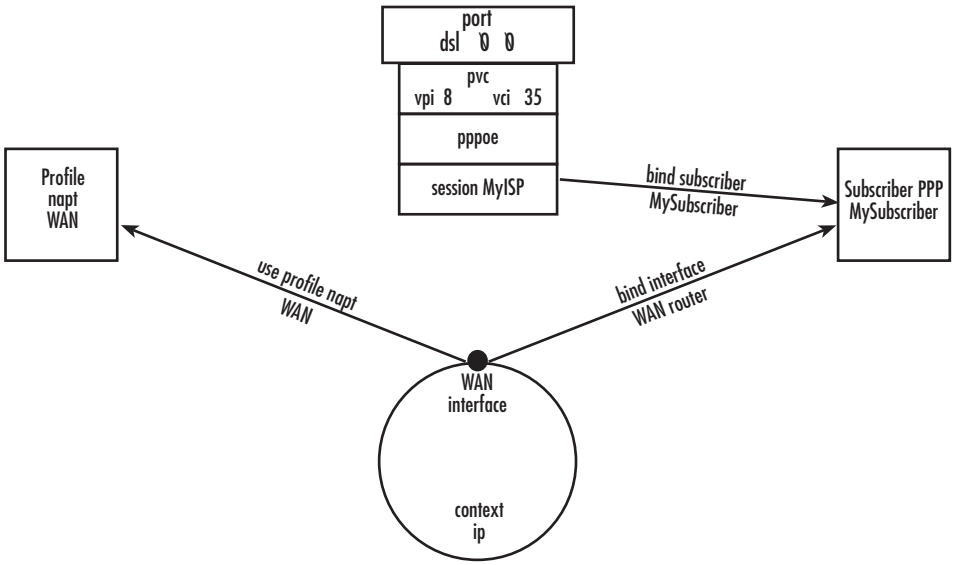


Figure 36. Configuring the G.SHDSL card for PPPoE

CAUTION

The Modem setup uses IP messages within its own subnet: 192.0.2.0/24. SmartNodes with built-in modems cannot use this subnet in any other way.

Line Setup

There is no line modulation setting. The modems automatically adapt to the bit rate and modulation used. The status LED on the back of the device is blinking while the modem attempts to connect and lit when the link is established. If the modem keeps blinking, check the cabling,

Configuring PPPoE

Figure 36 explains how to configure PPPoE on the SmartNode's built-in G.SHDSL card. To configure the DSL port for PPPoE, first you need to log in to the SmartNode via the CLI and enter configuration mode.

```
login: administrator
password: <enter>
node>enable
node>#configure
```

Next, you will need to create a WAN profile, create a WAN interface, and create a subscriber. Then, you can configure the DSL port (port dsl 0 0) for PPPoE.
Follow this example:

```
profile napt WAN

context ip router
   interface WAN
       ipaddress unnumbered
       point-to-point
       use profile napt WAN
       tcp adjust-mss rx mtu
       tcp adjust-mss tx mtu

subscriber ppp MySubscriber
  dial out
  authentication chap
  identification outbound <username> password <password>
  bind interface WAN router

port dsl 0 0
  pvc vpi 8 vci 35
     pppoe
        session MyISP
           bind subscriber MySubscriber
           no shutdown
```

The line - `use profile napt WAN` – defines that the NAPT profile <*profile*> will be used on the ip interface <*name*>. For PPPoE, you will only use outbound for identification. You will want to use authentication, which is why you bind to a subscriber. You can use authentication chap or authentication pap. The line - `bind subscriber MySubscriber` - binds the PPPoE session to the PPP subscriber, in case authentication is required. If you do not use authentication, then you will not have a subscriber and you will bind directly to the interface.

Configuration Summary

The modems offer multiple bridged Ethernet connections through logical channels within the DSL link. A logical connection is called a Permanent Virtual Circuit (PVC) and is identified by a VPI/VCI number pair. Consult your provider's configuration instructions for connections used on your DSL link. You define those PVCs inside "port dsl 0 0":

```
port dsl 0 0
  pvc vpi 8 vci 35
```

Iin the mode "pvc", you define what to do with the bridged Ethernet connection it offers:

- Bind one or more IP interfaces when your providers uses fixed ip addresses or DHCP in the network
- Enter PPPoE mode and define a PPP session if the provider is using PPPoE.

Note PPPoA is not supported.

Setting up permanent virtual circuits (PVC)

The modems currently available are using ATM to multiplex traffic over the DSL framing connection. ATM allows you to have separate logical connections running in parallel. Those connections are called permanent virtual circuits (PVC). All permanent virtual circuits use AAL5 framing.

Table 8. PVC Commands

	Command	Purpose
Step 1	**node(prt-dsl)[0/0]# [no] pvc vpi 8 vci 35**	Creates PVC 8/35 and enters configuration mode for this PVC. The "no"-variant deletes the PVC configuration.
Step 2	**node(pvc)[8/35]# encapsulation {llc|vc}**	Sets the encapsulation to be used. Optionally select either LLC encapsulation or VC multiplexing for this PVC. Default: llc

Using PVC channels in bridged Ethernet mode

The PVC offers a bridged Ethernet connection as specified in RFC1483, which can be used as an IP link e.g. with DHCP to assign the address, DNS server, and default gateway. To do this, you bind an IP interface to the PVC like it would be done to a normal Ethernet port.

Table 9. PVC channels in bridged Ethernet mode

	Command	Purpose
Step 1	**node(pvc)[vpi/vci]# [no] bind interface <if-name>**	Associates an IP interface configuration with this PVC.

Using PVC channels with PPPoE

The RFC1483 bridged Ethernet connection can also be used for PPPoE. To do this, you enter PPPoE mode within the PVC mode. All PPPoE commands apply as if the PVC was a regular Ethernet port.

Table 10. PVC channels in PPPoE mode

	Command	Purpose
Step 1	**node(pvc)[vpi/vci]# pppoe**	Enters PPPoE configuration mode for this PVC.
Step 2	**node(pppoe)# session <name>**	Defines a PPPoE session.
Step 3	**node(session)[<name>]# bind subscriber <subscriber-name>**	Links the session to a subscriber definition.
Step 4	**node(session)[<name>]# no shutdown**	Enables the PPPoE session

Note The bridged PVC connections are internally mapped to VLANs on a virtual Ethernet port 0/2. You will therefore see references to this third Ethernet port when displaying PPPoE status information or debug logs.

Diagnostics

Table 11. Diagnostics commans

	Command	Purpose
Step 1	**node> show dsl type**	Displays the type of modem installed.
Step 2	**node> show dsl line-state**	Displays information about the state of the DSL link.
Step 3	**node> show dsl version**	Display firmware version information for the modem.
Step 4	**node# debug dsl-setup**	Lists the configuration interactions between the gateway and the modem module.

Troubleshooting DSL Connections

Link State:

- Verify that the DSL link is established (status LED is continuously on)

PPPoE access:

- Check if "show pppoe detail 3" shows "State: .... opened". This indicates that the PVC is valid and a that you reached a PPPoE server through it.
- Check if "show ppp networks detail 3" shows "State: .... opened" for both the "LCP" and the "CHAP" section. If LCP is not working, there is probably no compatible authentication protocol configured. Make sure "authentication chap" and "authentication pap" are included in the subscriber setup. If only CHAP failed there may be an error with the username or password.
- Run the “debug” command: **node# debug dsl-setup** (See table 11 above).

Chapter 22 Basic IP routing configuration

Chapter contents

Introduction 236
Routing tables 236
Static routing 236
Policy routing 236
Basic IP routing configuration task list 236
Configuring static IP routes 237
Deleting static IP routes 238
Displaying IP route information 238
Configuring policy routing 239
Examples 240
Basic static IP routing example 240
Changing the default UDP port range for RTP and RTCP 241

Introduction

This chapter provides an overview of IP routing and describes the tasks involved in configuring static IP routing.

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 routing table stores routes to:

- Directly-attached interfaces or networks
- Static IP routes
- Routes learned dynamically from the Routing Information Protocol (RIP)

In the routing table, next-hop associations specify that a destination can be reached by sending packets to a next-hop router located on an optimal path to the destination. When the SmartNode receives an incoming packet, it checks the destination address, and attempts to associate this address with a next-hop address and outgoing interface. Routing algorithms must converge rapidly — i.e. all routers must agree on optimal routes. When a network event causes routes either to go down or to become unavailable, routers distribute routing update messages that permeate networks, causing recalculation of optimal routes that are eventually agreed upon by all routers. Routing algorithms that converge slowly can cause routing loops or network outages. Many algorithms can quickly select next-best paths and adapt to changes in network topology.

Static routing

Static routing involves packet forwarding on the basis of static routes configured by the system administrator. Static routes work well in environments where network traffic is relatively predictable and where the network topology is relatively simple. In contrast, dynamic routing algorithms adjust to changing network circumstances by analyzing incoming routing update messages. RIP uses dynamic routing algorithms.

Policy routing

IP routing makes decisions based on IP addresses. Policy Routing allows the user to configure IP routing based on more criteria than only the destination IP address. Within the IP Context, IP packets are categorized into traffic-classes which are used as a routing criterion. Three traffic-classes are defined—default, local-voice, and local-default. In addition packets can be categorized into user-defined traffic-classes by using ACL.

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 238)

- Displaying IP route information (see page 238)

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*.

```
node>enable
node#configure
node(cfg)#context ip router
node(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.

```
node>enable
node#configure
node(cfg)#context ip router
node(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.

```
node>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
```

Configuring policy routing

Step	Command	Purpose
1	***node*(cfg)#context ip router**	Enters the IP router context
2	**node(ctx-ip)[router]#[no] route destination netmask interface|gateway [metric] [traffic-class <traffic-class>]**	Define a static routing table entry

Where the syntax is:

- **destination**—The IP address of the target network or subnet.
- **Netmask—**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.
- **Interface|gateway**—the name of the outgoing interface to use for the target network or subnet, or the IP address of the outgoing interface
- **Metric**—(optional) 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.
- **Traffic class**—indicates that this static route is for IP traffic in the following <traffic-class>." If no traffic-class is specified, the routing table entry is of no traffic-class and is thus valid for packets of all traffic-classes.

Within IP context, IP packets are categorized into traffic-classes which are used as routing criteria. The following traffic-classes are defined:

ß Default:all IP packets that are arriving from the WAN or the LAN and need to be routed through.

ß Local-voice:IP packets that are created within the unit and contain voice data (RTP).

ß Local-default:IP packets that are created within the unit and do not contain voice, e.g., SIP signaling, DNS lookup, Telnet, etc.

In addition packets can be categorized into user-defined traffic-classes by using ACL.

A routing table entry may or may not have a traffic-class assigned. In the case that a routing table has no traffic-class assigned, it is valid for packets of all traffic-classes. On the other hand, if it does have a traffic-class assigned, the route is valid is restricted for packets of that given traffic-class.

Consider the following simple routing table example:

```
--------------------------------------------------------------------------
V  Destination       TrafficClass   Nexthop       Protocol  Metric  Flags
*  172.16.32.0/24                    eth1          local       1     UL
*  127.0.0.0/8                     loopback        local       1     UL
*  0.0.0.0/0        local-voice    172.16.32.1     static      0     UDG
*  0.0.0.0/0                       172.16.32.2     static      0     UDG
```

In this routing table two default routes (0.0.0.0/0) are defined. The first default route is valid for packets of the class local-voice only. The second default route is valid for all packets. Thus voice packets generated locally (traffic-class local-voice) will travel via the gateway (Nexthop) 172.16.32.1. All other packets will travel via the gateway (Nexthop) 172.16.32.2.

NOTE: If the second default route was missing, there would be no default route for packets of traffic-class other than local-voice.

The following modified commands are used with policy routing:

Route—refer to the 'route' command in the subsection "Configuring static IP routes" on page 237 in this chapter.

Ping—refer to the 'ping' command described in the subsection "Testing connections with the ping command" on page 127 in Chapter 10: IP interface configuration.

Traceroute—refer to the 'traceroute' command described in the subsection "Traceroute" on page 130 in Chapter 10: IP interface configuration.

Examples

Basic static IP routing example

Figure 37 shows an Internetwork consisting of three routers, a SmartNode device in the middle, and the four autonomous networks, with network addresses 10.1.5.0/16, 172.16.40.0/24, 172.17.100.0/24, and 10.2.5.0/16. The SmartNode shall be configured for the following IP routing scenario:

All packets for the Workstation with IP address 10.1.5.10 shall be forwarded to the next-hop router *Calvin*. All packets for network 10.2.5.0/16 shall be forwarded to the next-hop router *Hobbes*.

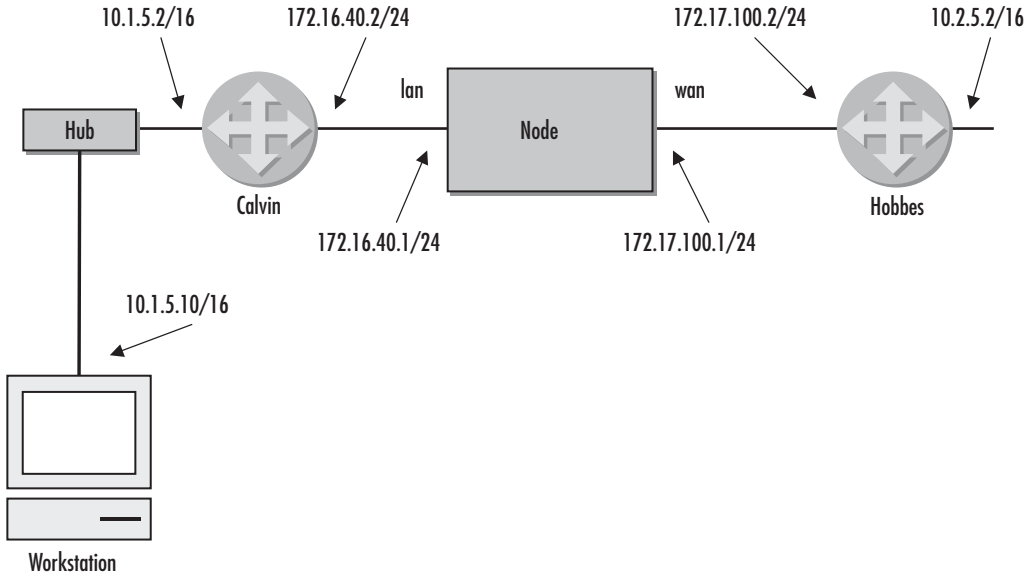


Figure 37. Internetwork with three routers and four networks

The necessary routing-table entries for the scenario described are listed below.

```
node>enable
node#configure
node(cfg)#context ip router
node(ctx-ip)[router]# route 10.1.5.10 255.255.255.255 172.16.40.2
node(ctx-ip)[router]# route 10.2.0.0 255.255.0.0 172.17.100.2
node>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 23 RIP configuration

Chapter contents

Introduction 243
Routing protocol 243
RIP configuration task list 244
- Enabling send RIP 244
- Enabling an interface to receive RIP 245
- Specifying the send RIP version 245
- Specifying the receive RIP version 246
- Enabling RIP learning 246
- Enabling an interface to receive RIP 247
- Enabling RIP announcing 247
- Enabling RIP auto summarization 248
- Specifying the default route metric 248
- Enabling RIP split-horizon processing 249
- Enabling the poison reverse algorithm 249
- Enabling holding down aged routes 250
- Setting the RIP route expiry 250
- Displaying RIP configuration of an IP interface 251
- Displaying global RIP information 251

Introduction

This chapter provides an overview of the Routing Information Protocol (RIP) and describes the tasks involved in configuring RIP features includes the following sections:

- Routing protocol
- RIP configuration task list (see page 244)

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. SmartNodes can send routing information updates every 30 seconds, which is termed *advertising*. If a router does not receive an update from another router for 180 seconds or more, it marks the routes served by the non-updating router as being unusable. If there is still no update after 240 seconds, the router removes all routing table entries for the non-updating router.

The metric that RIP uses to rate the value of different routes is the *hop count*. The hop count is the number of routers that can be traversed in a route. A directly connected network has a metric of zero; an unreachable network has a metric of 16. This small range of metrics makes RIP an unsuitable routing protocol for large networks

A SmartNode that is running RIP can receive a default network via an update from another router that is running RIP, or the router can source (generate) the default network itself with RIP. In both cases, the default network is advertised through RIP to other RIP neighbors.

a SmartNode 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 245)
- Specifying the send RIP version (see page 245)
- Specifying the receive RIP version (see page 246)
- Enabling RIP learning (see page 246)
- Enabling an interface to receive RIP (see page 247)
- Enabling RIP announcing (see page 247)
- Enabling RIP auto summarization (see page 248)
- Specifying the default route metric (see page 248)
- Enabling RIP split-horizon processing (see page 249)
- Enabling the poison reverse algorithm (see page 249)
- Enabling holding down aged routes (see page 250)
- Displaying RIP Configuration of an IP interface (see page 251)
- Displaying global RIP information (see page 251)

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*.

```
node(cfg)#context ip router
node(ctx-ip)[router]#interface wan
node(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*.

```
node(cfg)#context ip router
node(ctx-ip)[router]#interface wan
node(if-ip)[wan]#rip receive
```

Specifying the send RIP version

By default, RIP 1compatible packets are sent. 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*.

```
node(cfg)#context ip router
node(ctx-ip)[router]#interface wan
node(if-ip)[wan]#rip send version 1compatible
```

Specifying the receive RIP version

By default, RIP version 1 and version 2 packets are received. 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*.

```
node(cfg)#context ip router
node(ctx-ip)[router]#interface wan
node(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. Two RIP learning mechanisms are offered, 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*.

```
node(cfg)#context ip router
node(ctx-ip)[router]#interface wan
node(if-ip)[wan]#rip learn host
node(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*.

```
node(cfg)#context ip router
node(ctx-ip)[router]#interface lan
node(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 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*.

```
node(cfg)#context ip router
node(ctx-ip)[router]#interface wan
node(if-ip)[wan]#rip announce default
node(if-ip)[wan]#rip announce host
```

Enabling RIP auto summarization

Summarizing routes in RIP Version 2 improves scalability and efficiency in large networks.

Auto-summarization attempts to automatically summarize groups of adjacent routes into single entries, the goal being to reduce the total number of entries in the RIP routing table, reducing the size of the table and allowing the router to handle more routes.

RIP auto-summarization (automatic network number summarization) is disabled by default. With auto-summarization, the SmartNode summarizes sub prefixes to the Class A, Class B, and Class C network boundary when class network boundaries are crossed.

This procedure describes how to enable RIP auto-summarization on an interface

Mode: Interface

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

Example: Enabling RIP auto-summarization

The following example shows how to enable auto-summarization on IP interface wan.

```
node(cfg)#context ip router
node(ctx-ip)[router]#interface wan
node(if-ip)[wan]#rip auto-summary
```

Specifying the default route metric

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

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

Because metrics cannot be directly compared, you must specify the default metric in order to designate the cost of the redistributed route used in RIP updates. All routes that are redistributed will use the default metric.

Setting the default route metric, which is a number, indicating the distance to the destination network element, e.g. another router or SmartNode in a network, is possible with the **rip default-route-value** command. The value is between 1 and 15 for a valid route, or 16 for an unreachable route.

This procedure describes how to set the routing metric on an interface

Mode: Interface

Step	Command	Purpose
1	***node*(if-ip)[*name*]#rip default-route-value** *value*	Sets the routing metric to *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.

```
node(cfg)#context ip router
node(ctx-ip)[router]#interface wan
node(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*.

```
node(cfg)#context ip router
node(ctx-ip)[router]#interface wan
node(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*.

```
node(cfg)#context ip router
node(ctx-ip)[router]#interface wan
node(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*.

```
node(cfg)#context ip router
node(ctx-ip)[router]#interface wan
node(if-ip)[wan]#rip route-holddown
```

Setting the RIP route expiry

The **rip route-expiry** command sets the route expiry timeout for routes learned from RIP.

Mode: interface ip

Step	Command	Purpose
1	***node*(if-ip)[*name*]#rip route-expiry [1...3600]**	Sets the RIP route expiry timeout. Default: 180 seconds

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*.

```
node(cfg)#context ip router
node(ctx-ip)[router]#interface wan
node(if-ip)[wan]#show rip interface wan
Interface wan (IP context router):
--------------------------------------------------
                   listen: disabled
                   supply: enabled
             send version: 1compatible
          receive version: 1or2
               learn host: disabled
            learn default: disabled
            announce host: disabled
          announce static: disabled
         announce default: disabled
 announce self-as-default: disabled
           route-holddown: enabled
           poison-reverse: disabled
             auto-summary: disabled
            split-horizon: disabled
      default-route-value: 0
--------------------------------------------------
```

Displaying global RIP information

SmartWare also support displaying global RIP information for the IP router context. This procedure describes how to display the global RIP information

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#show rip**	Displays the RIP information

Example: Displaying global RIP information

The following example shows how to display the global RIP information.

```
node(cfg)#show rip
RIP information:
```

```
rip enabled
```

Chapter 24 Access control list configuration

Chapter contents

Introduction 254
About access control lists 254
What access lists do 254
Why you should configure access lists 254
When to configure access lists 255
Features of access control lists 255
Access control list configuration task list 256
Mapping out the goals of the access control list 256
Creating an access control list profile and enter configuration mode 257
Adding a filter rule to the current access control list profile 257
Adding an ICMP filter rule to the current access control list profile 259
Adding a TCP, UDP or SCTP filter rule to the current access control list profile 261
Binding and unbinding an access control list profile to an IP interface 263
Displaying an access control list profile 264
Debugging an access control list profile 264
Examples 266
Denying a specific subnet 266

Introduction

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

This chapter includes the following sections:

- About access control lists
- Access control list configuration task list (see page 256)
- Examples (see page 266)

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 38 host A is allowed to access the Human Resources network and host B is prevented from accessing the Human Resources network.

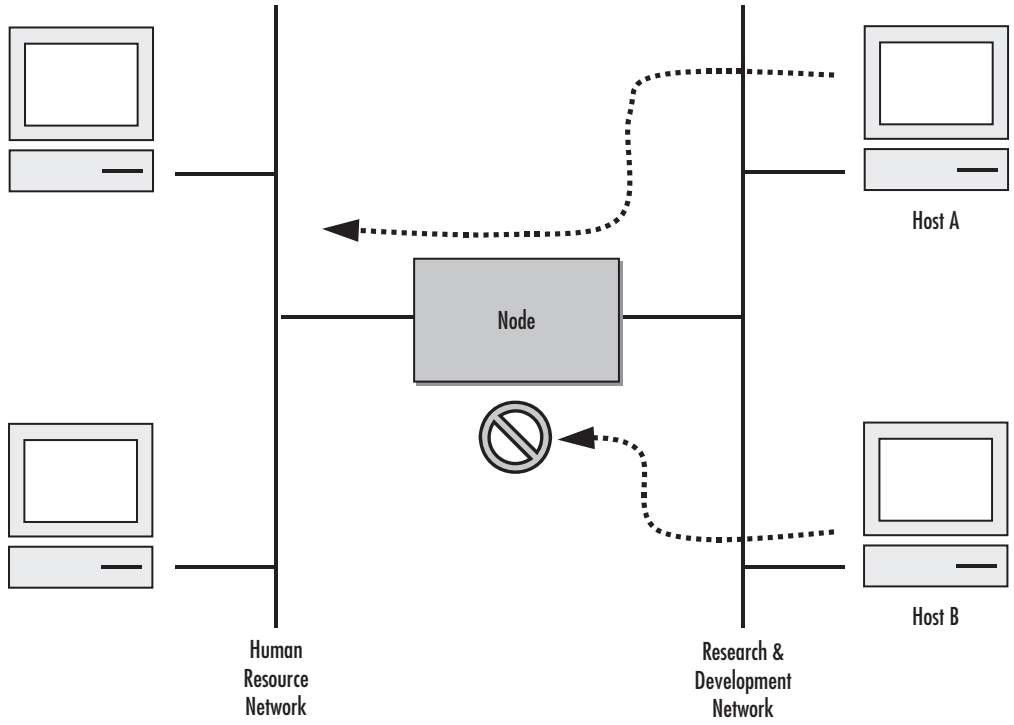


Figure 38. Using traffic filters to prevent traffic from being routed to a network

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 257)
- Adding a filter rule to the current access control list profile (see page 257)
- Adding an ICMP filter rule to the current access control list profile (see page 259)
- Adding a TCP, UDP or SCTP filter rule to the current access control list profile (see page 261)
- Binding and unbinding an access control list profile to an IP interface (see page 263)
- Displaying an access control list profile (see page 264)
- Debugging an access control list profile (see page 264)

Mapping out the goals of the access control list

To create an access control list you must:

- Specify the protocol to be filtered
- Assign a unique name to the access list
- Define packet-filtering criteria

A single access control list can have multiple filtering criteria statements.

Before you begin to enter the commands that create and configure the IP access control list, be sure that you are clear about what you want to achieve with the list. Consider whether it is better to deny specific accesses and permit all others or to permit specific accesses and deny all others.

Note Since a single access control list can have multiple filtering criteria statements, but editing those entries online can be tedious. Therefore, we recommend editing complex access control lists offline within a configuration file and downloading the configuration file later via TFTP to your SmartNode device.

Creating an access control list profile and enter configuration mode

This procedure describes how to create an IP access control list and enter access control list configuration mode

Mode: Administrator execution

Step	Command	Purpose
1	***node*(cfg)#profile acl** *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.

```
node>enable
node#configure
node(cfg)#profile acl WanRx
node(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 13, "Link scheduler configuration" on page 151.
group	CoS group name.

If you place a *deny ip any any* rule at the top of an access control list profile, no packets will pass regardless of the other rules you defined.

Example: Create IP access control list entries

Select the access-list profile named WanRx and create some filter rules for it.

```
node(cfg)#profile acl WanRx
node(pf-acl)[WanRx]#permit ip host 62.1.2.3 host 193.14.2.11 cos Urgent
node(pf-acl)[WanRx]#permit ip 62.1.2.3 0.0.255.255 host 193.14.2.11
node(pf-acl)[WanRx]#permit ip 97.123.111.0 0.0.0.255 host 193.14.2.11
node(pf-acl)[WanRx]#deny ip any any
node(pf-acl)[WanRx]#exit
node(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 13, "Link scheduler configuration" on page 151.
group	CoS group name.

If you place a *deny ip any any* rule at the top of an access-list profile, no packets will pass regardless of the other rules you defined.

Example: Create ICMP access control list entries

Select the access-list profile named WanRx and create the rules to filter all ICMP echo requests (as used by the ping command).

```
node(cfg)#profile acl WanRx
node(pf-acl)[WanRx]#deny icmp any any type 8 code 0
node(pf-acl)[WanRx]#exit
node(cfg)#
```

The same effect can also be obtained by using the simpler message name option. See the following example.

```
node(cfg)#profile acl WanRx
node(pf-acl)[WanRX]#deny icmp any any msg echo
node(pf-acl)[WanRX]#exit
node(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 13, "Link scheduler configuration" on page 151.
cos-rtp	Optional. Specifies that the rule is intended to filter RTP/RTCP packets. In this mode you can specify different CoS groups for data packets (even port numbers) and control packets (odd port numbers). Note: this option is only valid when protocol UDP is selected.
group	CoS group name.
group-data	CoS group name for RTP data packets. Only valid when the rtp option has been specified
group-ctrl	CoS group name for RTCP control packets. Only valid when the rtp option has been specified.

Example: Create TCP, UDP or SCTP access control list entries

Select the access-list profile named WanRx and create the rules for:

Permitting any TCP traffic to host 193.14.2.10 via port 80, and permitting UDP traffic from host 62.1.2.3 to host 193.14.2.11 via any port in the range from 1024 to 2048.

```
node(cfg)#profile acl WanRx
node(pf-acl)[WanRx]#permit tcp any host 193.14.2.10 eq 80
node(pf-acl)[WanRx]#permit udp host 62.1.2.3 host 193.14.2.11 range 1024 2048
node(pf-acl)[WanRx]#exit
node(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.

```
node(cfg)#context ip router
node(cfg-ip)[router]#interface wan
node(cfg-if)[wan]#use profile acl WanRx in
```

Unbind an access control list profile from an interface.

```
node(cfg)#context ip router
node(cfg-ip)[router]#interface wan
node(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.

```
node#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.

```
node(cfg)#context ip router
node(cfg-ip)[router]#interface wan
node(cfg-if)[wan]#debug acl in 7
```

The following example enables the debug monitor for access control lists globally.

```
node#debug acl
```

The following example disables the debug monitor for access control lists globally.

```
node#no debug acl
```

Examples

Denying a specific subnet

Figure 39 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*. 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*.

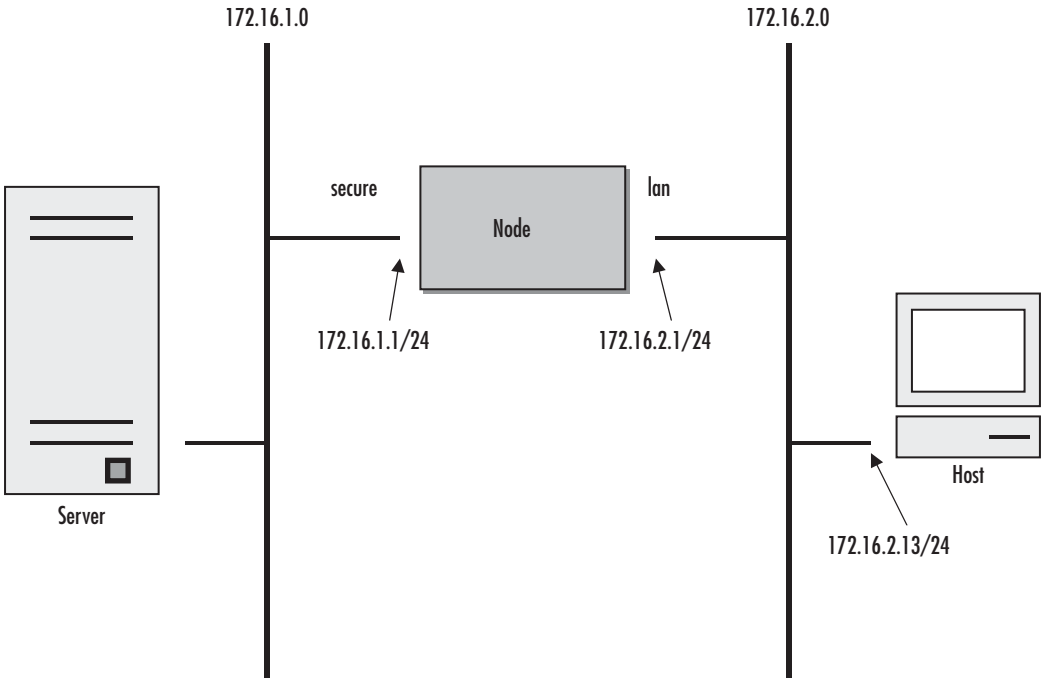


Figure 39. Deny a specific subnet on an interface

The commands that have to be entered are listed below.

```
172.16.2.1>enable
172.16.2.1#configure
172.16.2.1(cfg)#profile acl Jamming
172.16.2.1(pf-acl)[Jamming]#deny ip 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 25 SNMP configuration

Chapter contents

Introduction 268
Simple Network Management Protocol (SNMP) 268
SNMP basic components 268
SNMP basic commands 268
SNMP management information base (MIB) 269
Network management framework 269
Identification of a SmartNode via SNMP 269
SNMP tools 270
SNMP configuration task list 270
Setting basic system information 270
Setting access community information 272
Setting allowed host information 274
Specifying the default SNMP trap target 274
Displaying SNMP related information 275
Using the AdventNet SNMP utilities 275
Using the MibBrowser 276
Using the TrapViewer 277
Standard SNMP version 1 traps 279
SNMP interface traps 280

Introduction

This chapter provides overview information about Simple Network Management Protocol (SNMP) and describes the tasks used to configure those of its features supported.

This chapter includes the following sections:

- Simple Network Management Protocol (SNMP)
- SNMP tools (see page 270)
- SNMP configuration task list (see page 270)
- Using the AdventNet SNMP utilities (see page 275)
- Standard SNMP version 1 traps (see page 279)

Simple Network Management Protocol (SNMP)

The Simple Network Management Protocol (SNMP) is an application-layer protocol that facilitates the exchange of management information between network devices. It is part of the Transmission Control Protocol/Internet Protocol (TCP/IP) suite. SNMP enables network administrators to manage network performance, find and solve network problems, and plan for network growth.

SNMP basic components

An SNMP managed network consists of three key components: managed devices, agents, and network-management systems (NMSs).

A managed device is a network SN that contains an SNMP agent and resides on a managed network. Managed devices collect and store management information and make this information available to NMSs using SNMP. Managed devices, sometimes called network elements, can be routers and access servers, switches and bridges, hubs, computer hosts, or printers.

An agent is a network-management software module that resides in a managed device. An agent has local knowledge of management information and translates that information into a form compatible with SNMP.

An NMS executes applications that monitor and control managed devices. NMSs provide the bulk of the processing and memory resources required for network management. One or more NMSs must exist on any managed network.

SNMP basic commands

Managed devices are monitored and controlled using four basic SNMP commands: **read**, **write**, **trap**, and traversal operations.

- The **read** command is used by an NMS to monitor managed devices. The NMS examines different variables that are maintained by managed devices.
- The **write** command is used by an NMS to control managed devices. The NMS changes the values of variables stored within managed devices.
- The **trap** command is used by managed devices to asynchronously report events to the NMS. When certain types of events occur, a managed device sends a trap to the NMS.

- Traversal operations are used by the NMS to determine which variables a managed device supports and to sequentially gather information in variable tables, such as a routing table.

SNMP management information base (MIB)

A Management Information Base (MIB) is a collection of information that is organized hierarchically. MIBs are accessed using a network-management protocol such as SNMP. They are comprised of managed objects and are identified by object identifiers.

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using the subset of abstract syntax notation one (ASN.1) defined in the SMI. In particular, an *object identifier*, an administratively assigned name, names each object type. The object type together with an object instance serves to uniquely identify a specific instantiation of the object. For human convenience, a textual string, termed the descriptor, to refer to the object type, is often used.

An object identifier (OID) world-wide identifies a managed object in the MIB hierarchy. The MIB hierarchy can be depicted as a tree with a nameless root, the levels of which are assigned by different organizations.

Network management framework

This section provides a brief overview of the current SNMP management framework. An overall architecture is described in RFC 2571 “An Architecture for Describing SNMP Management Frameworks”. The SNMP management framework has several components:

- Mechanisms for describing and naming objects and events for the purpose of management. The first version, Structure of Management Information (SMIv1) is described in RFC 1155 “Structure and Identification of Management Information for TCP/IP-based Internets”, RFC 1212 “Concise MIB Definitions”, RFC 1213 “Management Information Base for Network Management of TCP/IP-based Internets: MIB-II”, and RFC 1215 “A Convention for Defining Traps for use with the SNMP”. The second version, SMIv2, is described in RFC 2233 “The Interfaces Group MIB using SMIv2”, RFC 2578 “Structure of Management Information Version 2 (SMIv2)”, RFC 2579 “Textual Conventions for SMIv2”, and RFC 2580 “Conformance Statements for SMIv2”.
- Message protocols for transferring management information. The first version, SNMPv1, is described in RFC 1157 “A Simple Network Management Protocol (SNMP).” The second version, SNMPv2, which is not an Internet standards track protocol, is described in RFC 1901 “Introduction to Community-Based SNMPv2” and RFC 1906 “Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)”.
- Protocol operations for accessing management information. The first set of protocol operations and associated protocol data unit (PDU) formats is described in RFC 1157. The second set of protocol operations and associated PDU formats is described in RFC 1905 “Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)”.
- A set of fundamental applications described in RFC 2573 “SNMP Applications” and the view-based access control mechanism described in RFC 2575 “View-Based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)”.

Identification of a SmartNode via SNMP

All product models have assigned sysObjectID.

Refer to the getting started guide of your product, or see the MIB definition file (.my) for sysObjectIDs.

The SNMP agent running in SmartWare is SNMP version 1 (SNMPv1) compliant. SNMP version 2 (SNMPv2) and SNMP version 3 (SNMPv3) are not currently supported.

SNMP tools

Patton recommends the AdventNet MibBrowser, TrapViewer and other SNMP tools. Check the AdventNet Web server at http://www.adventnet.com for latest releases.

Refer to section "Using the AdventNet SNMP utilities" on page 275 for more detailed information on how to use these tools.

SNMP configuration task list

To configure SNMP, perform the tasks described in the following sections. The tasks in the first three sections are required; the tasks in the remaining sections are optional, but might be required for your application.

- Setting basic system information (required) (see page 270)
- Setting access community information (required) (see page 272)
- Setting allowed host information (required) (see page 274)
- Specifying the default SNMP trap target (optional) (see page 274)
- Displaying SNMP related information (optional) (see page 275)

Setting basic system information

The implementation of the MIB-II system group is mandatory for all systems. By default, an SNMP agent is configured to have a value for any of these variables and responds to get commands from a NMS.

The following MIB II panels should be set::

- sysContact
- sysLocation
- sysName

The system sysContact object is used to define the contact person, together with information on how to contact that person.

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

The name used for sysName should follow the rules for ARPANET host names. Names must start with a letter, end with a letter or digit, and have as interior characters only letters, digits, and hyphens. Names must be 63 characters or fewer. For more information, refer to RFC 1035.

This procedure describes how to set these MIB-II system group objects

Mode: Administrator execution

Step	Command	Purpose
1	***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.

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 40.

The procedure to use the SNMP MIB browser is:

- Enter the community string **public** into the Community field in the upper right corner of the window. For safety reasons each entered character is displayed with a “*”.
- Access any of the supported MIB system group object by using the **GetNext** button from the button bar of the window.

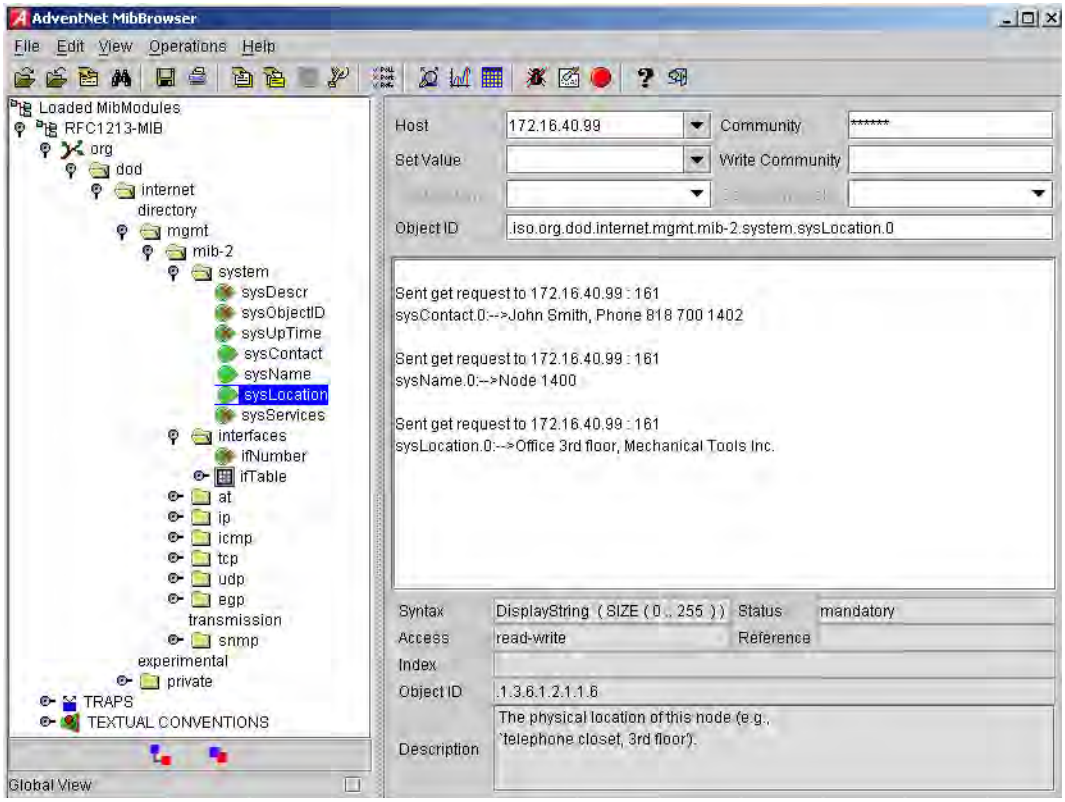


Figure 40. AdventNet MibBrowser displaying some of the System Group objects

Example: Setting the system group objects

In the following example the system information is set for later access via SNMP. See figure 40 for a typical MIB browser application accessing these MIB-II system group objects representing the system information.

```
node>enable
node#configure
node(cfg)#system contact "Bill Anybody, Phone 818 700 1504"
node(cfg)#system location "Wiring Closet, 3rd floor"
node(cfg)#system hostname "node"
(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.

```
node(cfg)#snmp community public ro
node(cfg)#snmp community Not4evEryOne rw
```

Note If no community is set on your SmartNode accessing any of the MIB objects is not possible!

Setting allowed host information

If a host has to access SNMP MIB objects on a certain node, it explicitly needs the right to access the SNMP agent. Therefore a host needs an entry, which allows accessing the device. The host is identified by its IP address and has to use a certain community string for security precautions.

Note The community which is to be used as security name to access the MIB objects has to be defined prior to the definition of allowed hosts.

This procedure describes adding a host that is allowed to access the MIB of this system

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#snmp host** *IP-address-of-SN* **security-name** *community*	Configures a host that with IP address *IP-address-of-SN* can access the MIB, 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 using community *public* as security name.

```
node(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 sends to a network management station. For example, an SNMP agent would send a trap when an interface's status has changed from up to down. The SNMP agent must know the address of the network management station so that it knows where to send traps. It is possible to define more than one SNMP trap target.

The SNMP message header contains a *community* field. The SNMP agent uses a defined community name, which is inserted in the trap messages header sent to the target. In most cases the target is a NMS, which only accepts a SNMP message header of a certain community.

This procedure describes how to define a SNMP trap target and enter community name

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#snmp target** *IP-address-of-SN* **security-name** *community*	Configures a SNMP trap target with IP-address-of-hostanme *SN* that receives trap messages 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.

```
node(cfg)#snmp target 172.16.224.44 security-name Not4evEryOne
```

Displaying SNMP related information

Displaying the SNMP related configuration settings is often necessary to check configuration modifications or when determining the behavior of the SNMP agent.

This procedure describes how to display information and configuration settings for SNMP

Mode: Configure

Step	Command	Purpose
1	***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.

```
node(cfg)#show snmp

SNMP Information:
  hostname : node
  location : Wiring Closet, 3rd floor
  contact  : Bill Anybody, Phone 818 700 1504

 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 following tools are the most useful:

- **MibBrowser**—used to view and operate on data available through a SNMP agent on a managed device
- **TrapViewer**—used to parse and view the received traps

The 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 41 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 SN. The AdventNet MibBrowser allows loading additional MIB files in the text format (the “my” file contains enterprise specific MIB definitions).

The right frame has labeled text fields to specify the basic parameters like host, community etc. and a Result text area display to view the results.

There are three ways in which the primary window of the MibBrowser can be viewed. It can be viewed with the result display, MIB description panel or multi-variable bind panel in the right frame. The view can be altered in three ways.

- The desired view can be set by the options provided in the display menu item under the view menu. (View → 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 41.

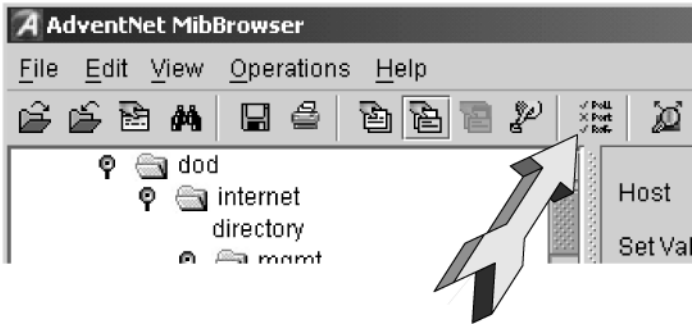


Figure 41. AdventNet MibBrowser Settings Button on the Toolbar

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. The TrapViewer can listen to one or more port at a time and the traps can be sent from any host.

Invoke the TrapViewer through the usage of the MibBrowser. To get to know more about the MibBrowser refer to section "Using the MibBrowser" on page 276. Figure 42 is a screen shot of the TrapViewer.

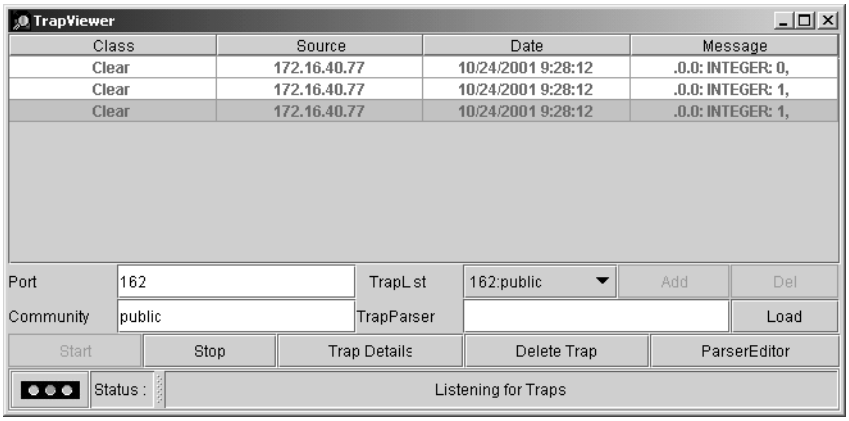


Figure 42. AdventNet TrapViewer displaying received traps

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.
- Click on *Add* button to add the port and community list on which the trap has to listen to. This is visible in the *TrapList* combo box.
- The port and community list can be deleted by clicking on the *Del* button.
- When you need to load a trap parser file, click on the *Load* button, which will open up a dialog box, from which you can load the parser file.
- In order to receive the traps now, click on the *Start* button. Upon clicking this button, TrapViewer begins to receive traps according to the as-specified port and community.
- Once received, the traps are listed in the trap table of the TrapViewer. By default, the trap table has the following four columns:
  - *Class* that defines the severity of the trap.
  - *Source* that displays the IP address of the source from where the traps were sent.
  - *Date* that shows the date and time when the trap was received.

- *Message* that by default has the object identifier format (sequence of numeric or textual labels on the SNs along a path from the root to the object) of the trap if any, or it is blank.

- The details of the traps can be viewed by clicking the *Trap Details* button or right click the trap in the trap table and select the option *View Trap Details*. Figure 43 show the screen of such a trap details window.

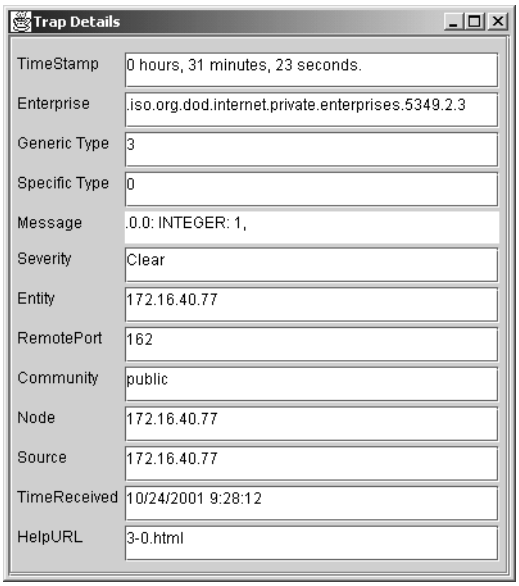


Figure 43. AdventNet Trap Details window of TrapViewer

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-warmStart, 2-linkDown, 3-linkUp, 4-authenticationFailure, 5-egpNeighborLoss. The trap type value 6 is identified as enterprise-specific value. This field shows the value based on the type of trap.
Specific Type	The specific trap type indicates the specific trap as defined in an enterprise-specific MIB. If the Generic type value is 6 then, this field shows a value greater than 0. If the generic type value is a value other than 6, then the field shows a value 0. This field can have values from 0 to 2147483647.
Message	This is a text field. By default, this field will always contain the Varbinds in the Trap PDU. This can be substituted with text.
Severity	This field shows the Severity or the intensity of the trap. They could be 0-All, 1-Critical, 2-Major, 3-Minor, 4-warning, 5-Clear and 6-info.

Table 12. Details available in the Trap Details window (Continued)

Trap Details	Description
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

The following standard SNMP version 1 traps are supported. The descriptions are taken from RFC 1215 “Convention for defining traps for use with the SNMP”.

```
warmStart TRAP-TYPE
ENTERPRISE snmp
DESCRIPTION
"A warmStart trap signifies that the sending protocol entity is reinitializing
itself such that neither the agent configuration nor the protocol entity implementa-
tion is altered."
::= 1

linkDown TRAP-TYPE
ENTERPRISE snmp
VARIABLES   { ifIndex }
DESCRIPTION
"A linkDown trap signifies that the sending protocol entity recognizes a failure in
one of the communication links represented in the agent's configuration."
::= 2
```

Note The linkDown trap is not sent if any of the ISDN ports has gone down.

```
linkUp TRAP-TYPE
ENTERPRISE snmp
VARIABLES   { ifIndex }
DESCRIPTION
"A linkUp trap signifies that the sending protocol entity recognizes that one of the
communication links represented in the agent's configuration has come up."
::= 3
```

Note The linkUp trap is not sent if any of the ISDN ports has come up.

```
authenticationFailure TRAP-TYPE
ENTERPRISE snmp
DESCRIPTION
"An authenticationFailure trap signifies that the sending protocol entity is the
addressee of a protocol message that is not properly authenticated. While implemen-
tations of the SNMP must be capable of generating this trap, they must also be capa-
ble of suppressing the emission of such traps via an implementation-specific
mechanism."
::= 4
```

Note The authenticationFailure trap is sent after trying to access any MIB object with a SNMP community string, which does not correspond to the system setting.

```
coldStart TRAP-TYPE
ENTERPRISE snmp
DESCRIPTION
"A coldStart trap signifies that the sending protocol entity is reinitializing
itself such that the agent's configuration or the protocol entity implementation may
be altered."
::= 0
```

Note The standard SNMP version 1 trap coldStart as listed below is *not* supported. After powering up, a warmStart trap message is sent if any trap target host is defined.

SNMP interface traps

The SmartNode sends Interface Traps (*linkUp*, *linkDown*) when the status of logical or physical interfaces change. Logical interfaces are interfaces defined in the IP context and CS context. Physical interfaces are ports.

The SmartNode adds an entry to event log for each Interface Traps it sends:

```
node(cfg)#show log

...
2002-09-06T14:54:35 : LOGINFO  : Link up on interface h323_60.
2002-09-06T14:54:35 : LOGINFO  : Link up on interface h323_30.
2002-09-06T14:54:35 : LOGINFO  : Link up on interface isdn20.
2002-09-06T14:54:38 : LOGINFO  : Link up on interface ETH00.
2002-09-06T14:54:38 : LOGINFO  : Link up on interface ETH01.
```

```
2002-09-06T14:54:39 : LOGINFO  : Link up on interface eth00.
2002-09-06T14:54:39 : LOGINFO  : Link up on interface eth01.
2002-09-06T14:56:02 : LOGINFO  : Link up on interface SLOT2:00 ISDN D
2002-09-10T14:21:20 : LOGINFO  : Link down on interface SLOT2:00 ISDN
...
```

Chapter 26 SNTP client configuration

Chapter contents

Introduction 283
SNTP client configuration task list 283
Selecting SNTP time servers 284
Defining SNTP client operating mode 284
Defining SNTP local UDP port 285
Enabling and disabling the SNTP client 286
Defining SNTP client poll interval 286
Defining SNTP client constant offset to GMT 287
Defining the SNTP client anycast address 287
Enabling and disabling local clock offset compensation 288
Showing SNTP client related information 289
Debugging SNTP client operation 289
Recommended public SNTP time servers 290
NIST Internet time service 290
Additional information on NTP and a list of other NTP servers 291

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 290)

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 284)
- Defining SNTP client operating mode (see page 284)
- Defining SNTP local UDP port (see page 285)
- Enabling and disabling the SNTP client (see page 286)
- Defining the SNTP client anycast address (see page 287)
- Defining SNTP client constant offset to GMT (see page 287)
- Enabling and disabling local clock offset compensation (see page 288)
- Defining SNTP client poll interval (see page 286)
- Showing SNTP client related information (see page 289)
- Debugging SNTP client operation (see page 289)

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.

```
node(cfg)#sntp-client server primary 172.16.1.10
node(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 287 for more details.

Example: Configuring SNTP client operating mode

Configures the SNTP client operating mode to unicast operation

```
node(cfg)#sntp-client operating-mode unicast
```

Configures the SNTP client operating mode to anycast operation

```
node(cfg)#sntp-client operating-mode anycast
```

Configures the SNTP client operating mode to multicast operation

```
node(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 SmartNode) 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 (SmartNode) 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

```
node(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

```
node(cfg)#sntp-client
```

Example: Disabling the SNTP client operation

```
node(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.

```
node(cfg)#sntp-client poll-interval 30
```

Defining SNTP client constant offset to GMT

Setting the offset of the 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!

Use the “clock local offset” command to configure the local clock offset.

This procedure describes how to display the local time.

Mode: Configure

Step	Command	Purpose
1	[*name*]**#show clock local**	Displays the local time, UTC and the offset of the local time from UTC.

This procedure describes how to use the **clock local offset** command.

Mode: Configure

Step	Command	Purpose
1	[*name*]**(cfg)#clock local offset (+|-)hh:mm**	Sets the offset from UTC to local time.

Defining the SNTP client anycast address

Anycast mode is designed for use with a set of cooperating servers whose addresses are not known beforehand. An anycast client sends a request to the designated local broadcast or multicast group address as described below. For this purpose, the NTP multicast group address assigned by the IANA is used. One or more anycast servers listen on the designated local broadcast address or multicast group address. Each anycast server, upon receiving a request, sends a unicast reply message to the originating client. The client then binds to the first such message received and continues operation in unicast mode. Subsequent replies from other anycast servers are ignored.

In anycast mode, the SmartNode sends a request to a designated local broadcast or multicast group address and expects a reply from one or more anycast servers. The SmartNode uses the first reply received to establish the particular server for subsequent unicast operations. Later replies from this server (duplicates) or any other server are ignored.

Other than the selection of address in the request, the operations of anycast and unicast clients are identical.

This procedure describes how to set local broadcast address or multicast group address to which the anycast request is sent

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#sntp-client anycast-address** *ip-address* **{port |** *port-number***}**	Set the anycast-address to *ip-address* a designated local broadcast or multicast group address to which a request is sent. In addition an explicit SNTP server *port-number* in the range from 1 to 65535 can be defined or the argument port is selected, which sets the value for port to 123. If none of the optional argument is used the value for port is set to 123.

Note This command is only relevant in *anycast operating-mode*.

Example: SNTP client anycast address

In the following example anycast requests are sent to SNTP server at IP address 132.163.4.101 using port 123 of the SNTP server.

```
node(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

```
node(cfg)#sntp-client root-delay-compensation
```

Example: Disabling the SNTP client root delay compensation

```
node(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

```
node(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.

```
node(cfg)#debug sntp client
node(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.

```
node(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.

Table 13. Time servers operated by NIST

Server Name	IP Address	Location
nist1.aol-va.truetime.com	205.188.185.33	DC/Virginia
utcnist.colorado.edu	128.138.140.44	Colorado
nist1.aol-ca.truetime.com	207.200.81.113	California
nist1-dc.glassey.com	216.200.93.8	DC/Virginia
nist1.datum.com	63.149.208.50	California
nist1-ny.glassey.com	208.184.49.9	New York City
nist1-sj.glassey.com	207.126.103.204	California
time-a.timefreq.bldrdoc.gov	132.163.4.101	Colorado
time-b.timefreq.bldrdoc.gov	132.163.4.102	Colorado
time-c.timefreq.bldrdoc.gov	132.163.4.103	Colorado

For more information about NIST Internet Time Service (ITS) check their web server at **http://www.boulder.nist.gov/timefreq/service/its.htm**

Additional information on NTP and a list of other NTP servers

The site http://ntp.isc.org contains a maintained list of available NTP/SNTP servers. Please only use the ones with an open access policy!

Chapter 27 DHCP configuration

Chapter contents

Introduction 293
DHCP-client configuration tasks 294
Enable DHCP-client on an IP interface 294
Release or renew a DHCP lease manually (advanced) 296
Get debug output from DHCP-client 296
DHCP-server configuration tasks 297
Configure DHCP-server profiles 297
Use DHCP-server profiles and enable the DHCP-server 299
Define the bootfile (Option 67) for the DHCP-server 300
Define the TFTP server (Option 66) for the DHCP-server 300
Check DHCP-server configuration and status 300
Get debug output from the DHCP-server 301
Configure DHCP-relay 302
Create/Modify DHCP-Relay profile 302
Enable/Disable DHCP-Relay Agent 303

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 294)
- DHCP-server configuration tasks (see page 297)

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.

DHCP-server and DHCP-client are illustrated in figure 44.

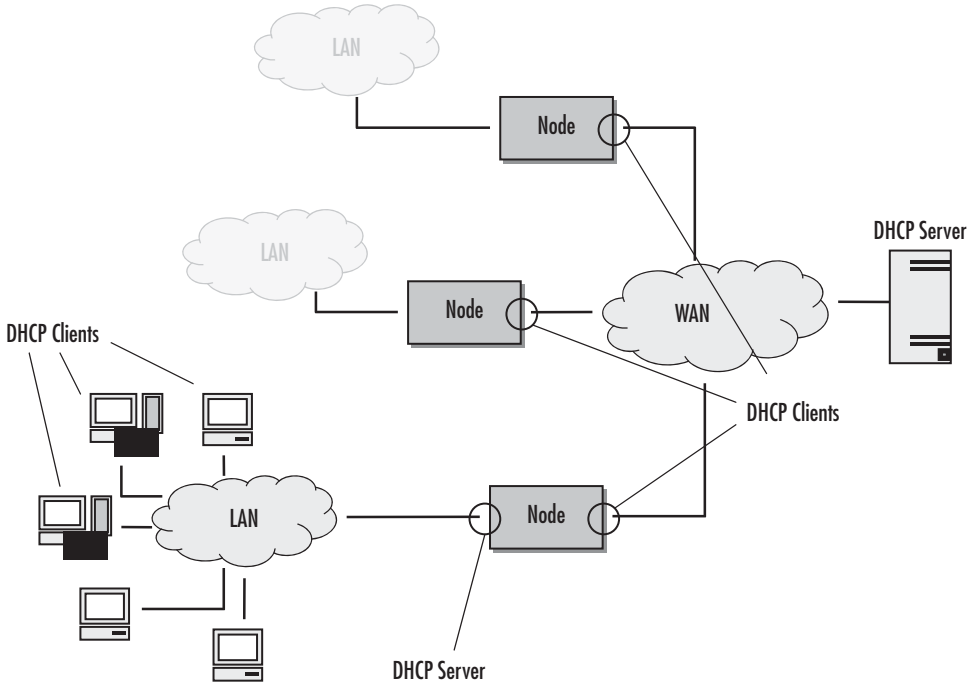


Figure 44. DHCP-client and DHCP-server

DHCP-client configuration tasks

To configure the SmartNode as DHCP-client perform the steps mentioned below.

- Enable DHCP-client on an IP interface
- Release or renew a DHCP lease manually (advanced) (see page 296)
- Get debug output from DHCP-client (see page 296)
- Configure DHCP agent

Enable DHCP-client on an IP interface

On every created IP interface a DHCP-client could be enabled. If enabled, the SmartNode gets the IP address for this interface from a DHCP-server. Additionally other configuration information is received for this IP

interface, e.g. the default gateway, DNS server IP addresses, etc. To enable the DHCP-client on an IP interface perform the steps described below.

Mode: context IP

Step	Command	Purpose
1	***node*(ctx-ip)[router]#interface** *name*	Creates an IP interface with name *name* and enters 'configure' configuration mode
2	***node*(if-ip)[*name*]#ipaddress dhcp**	Enables the DHCP-client on this IP interface. (See note)
3	***node*(if-ip)[*name*]#show dhcp-client**	Displays status information about the DHCP-client For example, default gateway, lease expire time, etc.

Note If you are connected to the SmartNode by Telnet over the IP interface on which you enable the DHCP-client, the connection is lost after entering the command **ipaddress dhcp**. You need to know the new IP address distributed from the DHCP-server to connect to the SmartNode again!

Example: Enable DHCP-client on an IP interface

```
node(cfg)#context ip
node(ctx-ip)[router]#interface eth0
node(if-ip)[eth0]#ipaddress dhcp
node(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 by Telnet over the IP interface on which you release the DHCP lease, the connection is lost after entering the command **dhcp-client release**. You need an other way (e.g. a serial connection) to connect to the SmartNode again and to enter the command **dhcp-client renew**!

Get debug output from DHCP-client

This procedure describes how to enable/disable DHCP-client debug monitor

Mode: Any

Step	Command	Purpose
1	***node*(if-ip)[*name*]#[no] debug dhcp-client**	Enables/disables the DHCP-client debug monitor

Example: Enable DHCP debug monitor

This example shows how to enable the DHCP-client debug monitor and the debug output of the command **dhcp-client release** and **dhcp-client renew**.

```
node(cfg)#context ip
node(ctx-ip)[router]#interface eth0
node(if-ip)[eth0]#debug dhcp-client
node(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.

node(if-ip)[eth0]#dhcp-client renew
01:17:46  DHCPC > router/eth0 (Init): Tearing down IP interface
01:17:46  DHCPC > router/eth0 (Init): Broadcasting DHCP discover (xid 0f839e56, secs
0).
01:17:46  DHCPC > router/eth0 (Init):  Requesting IP address 172.16.224.102
01:17:47  DHCPC > router/eth0 (Slct): Got offer from 172.16.1.10 for IP
172.16.224.102
01:17:47  DHCPC > router/eth0 (Slct): Selected offer for 172.16.224.102
01:17:47  DHCPC > router/eth0 (Slct): Broadcasting DHCP request (select) (xid
6ff42c38, secs 1).
```

```
2001-01-01T01:17:47 : LOGINFO    : router/eth0 (Rqst): Got DHCP lease for
172.16.224.102
01:17:47  DHCPC > router/eth0 (Rqst): DHCP ACK received.
01:17:47  DHCPC > router/eth0 (Rqst): Lease is valid for 28800 seconds
01:17:47  DHCPC > router/eth0 (Rqst):   (t1: 14400, t2: 25200)
01:17:47  DHCPC > router/eth0 (Rqst): Got DHCP lease for 172.16.224.102
01:17:47  DHCPC > router/eth0 (Rqst): Configuring IP interface
2001-01-01T01:17:48 : LOGINFO    : Link down on interface eth0.
2001-01-01T01:17:48 : LOGINFO    : Link up on interface eth0.
```

DHCP-server configuration tasks

To configure the SmartNode as DHCP-server perform the steps mentioned below.

- Configure DHCP-server profiles
- Use DHCP-server profiles and enable the DHCP-server (and to clear lease database) (see page 299)
- Define the bootfile for the DHCP-server (see page 300)
- Define the TFTP server for the DHCP-server (see page 300)
- Check DHCP-server configuration and status (see page 300)
- Get debug output from the DHCP-server (see page 301)
- Configure DHCP-relay (see page 302)

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, then 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.

```
node(cfg)#profile dhcp-server LAN
node(pf-dhcps)[lan]#network 192.168.1.0 255.255.255.0
node(pf-dhcps)[lan]#include 192.168.1.32 192.168.1.63
node(pf-dhcps)[lan]#lease 2 days
node(pf-dhcps)[lan]#default-router 192.168.1.1
node(pf-dhcps)[lan]#domain-name-server 80.254.161.125
node(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	**[***name***](ctx-ip)[***router***]#dhcp-server use profile <profile name>**	Chooses a DHCP profile. **profile:** Name of the DHCP profile to choose.
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.

```
node(ctx-ip)[router]#dhcp-server use LAN
node(ctx-ip)[router]#dhcp-server
```

Define the bootfile (Option 67) for the DHCP-server

The DHCP server can define bootfile (DHCP option 67). In contrast to the existing options bootfile and next-server, these options do not use BOOTP fields but DHCP options.

Mode: profile dhcp-server

Step	Command	Purpose
1	**[**name**](pf-dhcps)[profile]#[no] bootfile-opt67 <bootfile>**	Defines bootfile (option 67).

Define the TFTP server (Option 66) for the DHCP-server

The DHCP server can define tftp-server (DHCP option 66). In contrast to the existing options bootfile and next-server, these options do not use BOOTP fields but DHCP options.

Mode: profile dhcp-server

Step	Command	Purpose
1	**[**name**](pf-dhcps)[profile]#[no] tftp-server-opt66 <tftp-server>**	Defines TFTP server (option 66).

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:

```
node(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.

```
node(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
```

Configure DHCP-relay

SmartNodes support the DHCP-Relay functionality. To avoid having a DHCP-Server in every subnet, SmartNodes now can forward DHCP-Requests either directly to a DHCP-Server or to another DHCP-Relay. To configure the relay, there is a profile type **dhcp-relay.** The minimum configuration of such a profile only needs one IP where DHCP-Requests are sent to. Note that a SmartNode can be configured either as DHCP-Server or as DHCP-Relay, but not both in parallel.

Create/Modify DHCP-Relay profile

To use a SmartNode as DHCP-Relay Agent, a dhcp-relay profile must be created. Some things you should know about the Relay Agent profile:

- In order to create valid profile, a dhcp-relay profile must contain at least one destination IP.
- There is no limit of dhcp-relay profiles, but there is only one active profile at the time.
- Modification of the dhcp-relay profile currently in use can be made anytime. Changes are activated immediately. When all destination IPs are removed the profile is considered invalid and the DHCP-Relay is shut down.

Mode: Configure

Step	Command	Purpose
1	***[name]*(cfg) #[no] profile dhcp-relay** name	Enter dhcp-relay profile mode. A new profile is created if none with the given name exists.
2	***[name]* (pf-dhcpr)[name]#destination** <ip-address> *or* ***[name]* (pf-dhcpr)[name]#destination after** <index> <ip-address> *or* ***[name]* (pf-dhcpr)[name]#destination before** <index> <ip-address> *or* ***[name]* (pf-dhcpr)[name]#no destination** <index>	Add or remove an IP-Address to the profile. The IP-Address can be any kind of address, means uni-, mulit-, or broadcast. Be aware that every DHCP- Request will be relayed to all ip- addresses in the list in the order they are listed. The DHCP-Relay profile is only valid if it contains at least one destination IP. The maximum amount of destinations is limited to 16.
3	***[name]* (pf-dhcpr)[name]#max-hops <1...16>** *or* ***[name]* (pf-dhcpr)[name]#no max-hops**	Sets the max-hops a packet may have passed so far. When a packet exceeds the max-hops limit, it is silently discarded. Max-hops is by default set to 4. This is an optional configuration command.
4	***[name]* (pf-dhcpr)[name]#traffic-class** <traffic-class>	Sets traffic class for DHCP packets. The traffic class may be new or may already exist. This is an optional configuration command.

Enable/Disable DHCP-Relay Agent
Mode: Context IP

Step	Command	Purpose
1	***[name]* (ctx-ip)*[name]*# dhcp-relay use <profile>** *or* ***[name]* (ctx-ip)*[name]*# no dhcp-relay**	Enable or disable DHCP-Relay. DHCP-Relay cannot be enabled if a DHCP-Server is running.

Chapter 28 DNS configuration

Chapter contents

Introduction 305
DNS configuration task list 305
Enabling the DNS resolver 305
Enabling the DNS relay 306

Introduction

The domain name system (DNS) enables users to contact a remote host by using easily remembered text labels (www.patton.com, for example) instead of having to use the host's numeric address (209.45.110.15, for example). When DNS names are entered as part of configuration commands or CLI exec mode commands in applications like Ping, Traceroute, or Tftp, the SmartNode uses a DNS resolver component to convert the DNS names into the numeric address.

The SmartNode can be configured as a caching DNS relay server to speed data transfers, acting as the DNS server for a private network. In this configuration, hosts in the network send their DNS queries to the SmartNode, which checks to see if the DNS name is in its DNS resolver cache. If it finds the name in cache, the SmartNode uses the cached data to resolve the DNS name into a numeric IP address. If the name is not in cache, the query is forwarded on to a DNS server. When the SmartNode receives the answer from the server, it adds the name to the cache, and forwards it on to the host that originated the query. This process enables the SmartNode to provide answers more quickly to often-queried DNS names, reducing the number of DNS queries that must be sent across the access link.

DNS configuration task list

The following sections describe how to configure the DNS component:

- Enabling the DNS resolver
- Enabling the DNS relay

Enabling the DNS resolver

To enable the SmartNode DNS resolver, you must configure it with the address of one or more DNS servers that will be used to resolve DNS name queries. If multiple DNS servers are configured, the SmartNode will query each server in turn until a response is received. DNS servers are configured as follows:

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#dns domain-name server** *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 SmartNode DNS resolver and increase the size of the DNS cache to 100 entries.

```
node>enable
node#configure
node(cfg)#dns-client server 62.2.32.5
node(cfg)#dns-client server 62.2.100.45
node(cfg)#dns-client cache 100
```

You can test the DNS server configuration using the **dns-lookup** command as follows:

Example: Testing DNS server configuration

```
node(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:

```
node(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)
node(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.

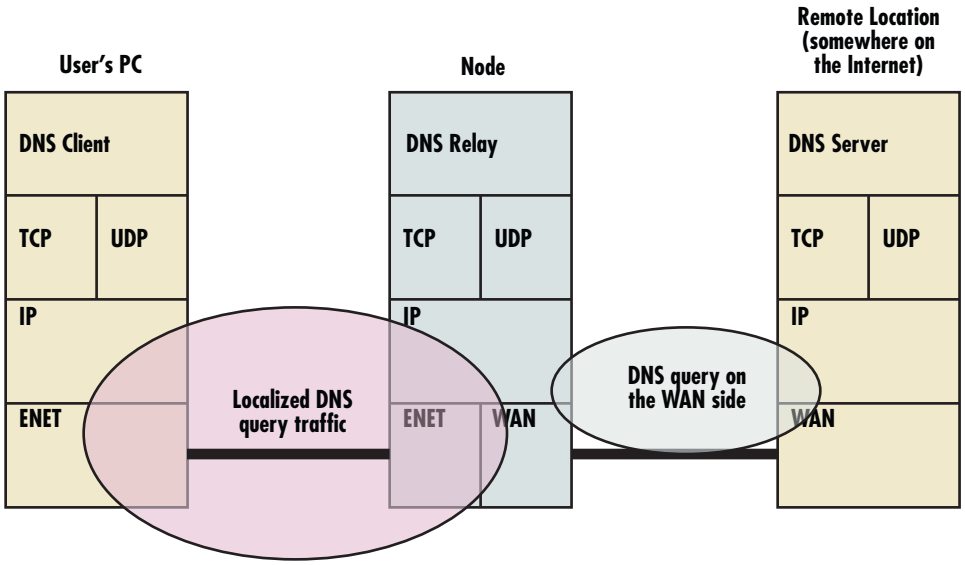


Figure 45. DNS relay diagram

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 45). 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 SmartNode 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 SmartNode which reduces congestion on the WAN side of the SmartNode
- Multiple DNS servers can be consulted from the SmartNode

The DNS resolver must be configured before you can use the DNS relay feature (see section "Enabling the DNS resolver" on page 305 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.

```
node>enable
node#configure
node(cfg)#dns-relay
```

Note If a DHCP server profile has been set up, you can announce the SmartNode as domain name server to the DHCP clients as follows:

```
node(cfg)#profile dhcp-server LAN
node(pf-dhcps)[LAN]#domain-name-server <ip-address>
```

Where *ip-address* must be the IP address of the SmartNode IP interface to which the DHCP clients are connected.

Chapter 29 DynDNS configuration

Chapter contents

Introduction 309
DynDNS configuration task list 309
Creating a DynDNS account 309
Configuring the DNS resolver 309
Configuring basic DynDNS settings 310
Configuring the DynDNS server 310
Configuring advanced DynDNS settings (optional) 311
Defining a mail exchanger for your hostname 311
Troubleshooting 311

Introduction

SmartNodes are often used in applications where the addresses of their IP interfaces are not assigned statically (i.e. permanently) but instead are configured dynamically. In these applications, the IP address is assigned dynamically using protocols like DHCP or PPP. The problem with dynamically assigning addresses is that when the IP address changes, remote devices can no longer contact the SmartNode because they do not know what the new address is.

Dynamic DNS (DynDNS) addresses this problem by registering a permanent hostname for your SmartNode. DynDNS then directs traffic sent to the registered host name on to the SmartNode's ISP-assigned dynamic IP address, enabling the SmartNode to be accessed from the Internet without knowing its current dynamic IP address.

The DNS server used for registration is operated by Dynamic Network Services, Inc. You can find detailed information about the company and the services it offers on the webpage www.dyndns.org. The company offers different levels of service. The basic services are offered free of charge, while the more advanced services are chargeable.

The SmartNode supports the following DynDNS services:

- Dynamic DNS
- Static DNS
- Custom DNS

DynDNS configuration task list

This section describes configuring the DynDNS service. All possible configurations, which are involved in a specific configuration topic are described in the respective configuration task. To get a minimal working configuration of the DynDNS client, you must execute all the configuration tasks of the list below, except the tasks explicitly marked as optional.

- Creating a DynDNS account
- Configuring the DNS resolver
- Configuring basic DynDNS settings
- Configuring advanced DynDNS settings (optional)

Creating a DynDNS account

Before using the DynDNS service, you must create a DynDNS account on the DynDNS server and add a hostname to your account, which can be updated by the SmartNode. Go to the DynDNS website at www.dyndns.org and follow the instructions on the webpage to create the account and add a hostname.

Configuring the DNS resolver

The DynDNS client requires that the SmartNode's DNS resolver be enabled. You can find additional information about how to configure the DNS resolver in chapter 28, "DNS configuration" on page 304.

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 SmartNode

Example: Configuring DynDNS

The following example shows the necessary steps required for a basic working configuration of the DynDNS client.

```
node>enable
node#configure
node(cfg)#context ip
node(ctx-ip)[router]#dyndns
node(dyndns)#authentication Bob 245gf46te
node(dyndns)#service dynamic
node(dyndns)#hostname myhostname.dyndns.org
node(dyndns)#observe eth1
```

Configuring the DynDNS server

A DynDNS server other than “dyndns.org” can be configured in case you or your provider runs an own DynDNS server.

Mode: context cs/dyndns

Step	Command	Purpose
1	[*name*]**(dyndns)# server [<server-name>]**	Set the DynDNS server name. The default value is “dyndns.org”.

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.

```
node>enable
node#configure
node(cfg)#context ip
node(ctx-ip)[router]#dyndns
node(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.

```
node(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.

```
node(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).

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 30 PPP configuration

Chapter contents

Introduction 314
PPP configuration task list 315
Creating an IP interface for PPP 315
Disable interface IP address auto-configuration from PPP 317
Creating a PPP subscriber 317
Trigger forced reconnect of PPP sessions using a timer 319
Disable interface IP address auto-configuration from PPP 319
Configuring a PPPoE session 319
Configuring PPP over a HDLC Link 321
Creating a PPP profile 321
Configuring the local and remote PPP MRRU 323
Displaying PPP configuration information 324
Debugging PPP 325
Sample configurations 329
PPP over Ethernet (PPPoE) 329
Without authentication, encapsulation multi, with NAPT 329
With authentication, encapsulation PPPoE 329
PPP over a HDLC Link (Serial Port) 330
Without authentication, numbered interface 330
With authentication, unnumbered interface 330
PPP over a HDLC Link (E1T1 Port) 330
Without authentication, numbered interface 330
PPP Dial-up over ISDN 331
PPP Dialer 331
Create a dialer 332
Create outbound destinations 332
Configure recovery strategy 333
Create inbound destinations 334
Debug dialer functionality 336
Example – Dial-on demand feature 336
Dial-up 337
Dial-up on demand 337
Dial-up on monitor 338
Dial-up nailed 338
..... 338

Introduction

This chapter describes how to configure the point-to-point protocol over different link layers.

The point-to-point protocol (PPP) provides a standard method for transporting multi-protocol datagrams over point-to-point links as defined by the RFC1661 etc. SmartWare offers PPP over the following link layers:

- PPP over Ethernet (PPPoE)
- PPP over HDLC

Figure 46 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 a HDLC Framed Serial Link are in the lower left corner. For PPP over ISDN, the elements are in the middle and the lower right corner.

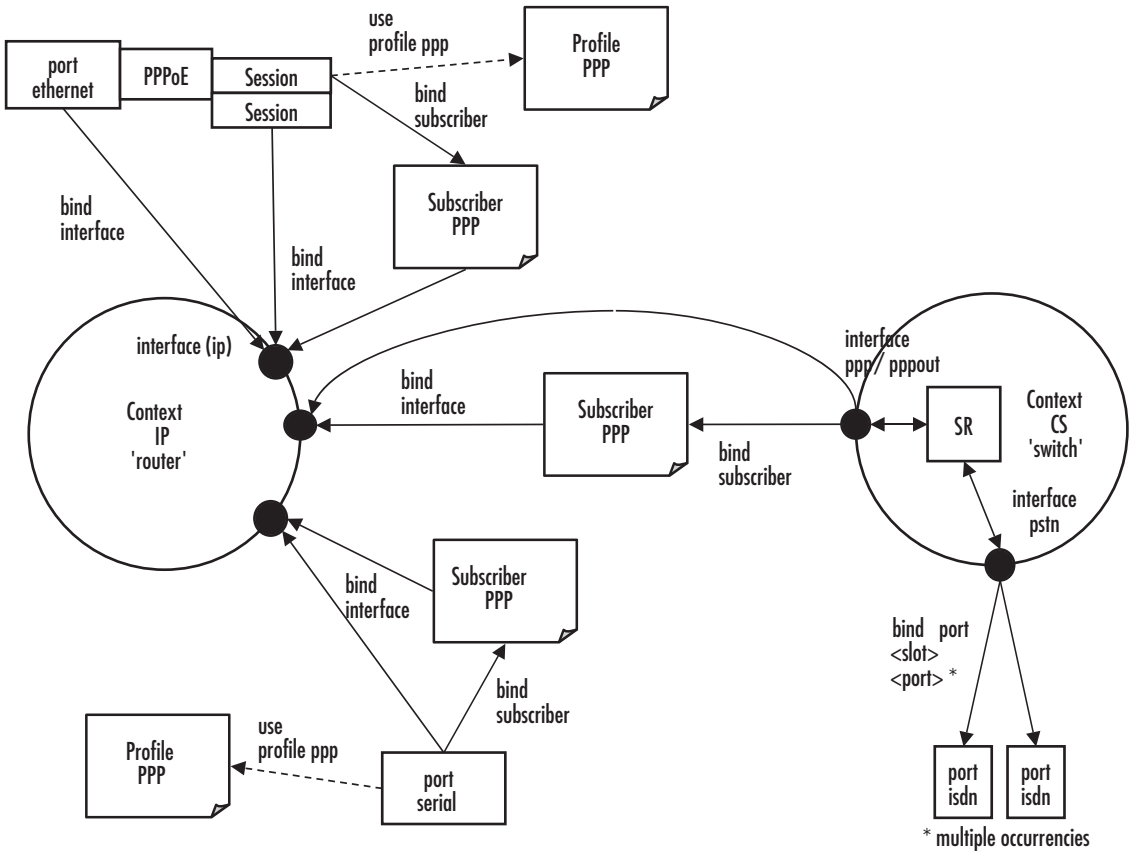


Figure 46. PPP configuration overview

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 317)
- Creating a PPP subscriber (for authentication) (see page 317)
- Configuring a PPPoE session (see page 319)
- Configuring PPP over a HDLC Link
- Creating a PPP interface within the CS context (not currently available)
- Creating a PSTN interface for PPP dial-in/dial-out (not currently available)
- Creating a PPP profile (see page 321)
- Displaying PPP configuration information (see page 324)
- Debugging PPP (see page 325)

Creating an IP interface for PPP

An IP interface is required to link a PPP connection to the IP context. The IP interface must apply a network address port translation (NAPT) if the PPP service provider only offers a single IP address and not an IP subnet, or if the IP addresses on the LAN shall be private and hidden behind a public IP address (see 11, "NAT/NAPT configuration" on page 132 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 11, "NAT/NAPT configuration" on page 132 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.

```
node(cfg)#context ip router
node(ctx-ip)[router]#interface ppp_interface
node(if-ip)[ppp_int~]#point-to-point
node(if-ip)[ppp_int~]#ipaddress unnumbered
node(if-ip)[ppp_int~]#tcp adjust-mss rx mtu
node(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.

Step	Command	Purpose
2	***node*(subscr)[name]# dial {in|out}**	Defines the direction of the connection establishment with PPP over ISDN. This information allows to use different subscribers for incoming and outgoing calls. With the other two link layers, set the direction as follows: • PPP over Ethernet: 'dial out' • PPP over Serial: 'dial in'
3	***node*(subscr)[name]# [no] authentication { (chap pap) | {chap|pap} }**	Defines the authentication protocol to be used, PAP and/or CHAP
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 315.

Example: Create a PPP subscriber

The procedure below creates a PPP subscriber for a PAP authentication with some Internet Service Provider.

```
node(cfg)#subscriber ppp joe_example
node(subscr)[joe_exa~]#dial out
node(subscr)[joe_exa~]#authentication pap
node(subscr)[joe_exa~]#identification outbound joeexample@isp.com password blue4you
node(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.

```
node(cfg)#port ethernet 0 0
node(prt-eth)[0/0]#encapsulation pppoe
node(prt-eth)[0/0]#no shutdown
node(prt-eth)[0/0]#pppoe
node(pppoe)[0/0]#session green
node(session)[green]#bind subscriber joe_example
node(session)[green]#no shutdown
```

Configuring PPP over a HDLC Link

This procedure describes how to configure PPP over a HDLC link. Different kind of physical ports can be configured for HDLC framed data transmission. On some ports the hdlc mode must be explicitly enabled (PRI, BRI), other ports have a HDLC framed nature (Serial). That means, PPP can be configured in different configuration modes. For this reason, the command description below refer always to the configuration mode in which ppp has been enabled by setting the encapsulation to 'ppp'. This configuration mode is called here 'hdlc-sub' but it is only an alias for the real mode.

Mode: hdlc-sub

Step	Command	Purpose
1	***node*(hdlc-sub)#[no] encapsulation ppp**	Enables/Disables PPP
3	***node*(hdlc-sub)#[no] bind interface** *name* **[router]** or ***node*(hdlc-sub)#[no] bind subscriber** *name* or ***node*(hdlc-sub)#[no] bind subscriber authentication { chap pap | { chap | pap } }**	Binds the HDLC link directly to the IP interface *name* in case no authentication is required Binds the HDLC link 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 HDLC link to the set of all PPP subscribers.
4 (optional)	***node*(hdlc-sub)#[no] use profile ppp *name***	Assigns a PPP profile other than the default profile.

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.

```
node(cfg)#profile ppp PPPoE
node(pf-ppp)[PPPoE]#mtu min 68 max 1492
node(pf-ppp)[PPPoE]#mru min 68 max 1492
node(pf-ppp)[PPPoE]#van-jacobson compression
node(pf-ppp)[PPPoE]#port ethernet 0 0
node(prt-eth)[0/0]#pppoe
node(pppoe)[0/0]#session green
node(session)[green]#use profile ppp PPPoE
```

Configuring the local and remote PPP MRRU

The PPP Maximum Receive Reconstructed Unit (MRRU) denotes the maximum reassembled MRU we are able to receive above the multi-link PPP protocol, i.e. this is the MRU after reassembling frames from individual links inside a multi-link bundle. The valid range and default value of MRRU can be configured in the PPP profile.

Mode: profile ppp

Step	Command	Purpose
1	[*name*] **(pf-ppp)**[*profile*]**# mrru [min <min>] [max <max>] [default <default>]**	Configures the minimum, maximum and default value of the local MRRU. The configured range (min..max) is forced during PPP LCP negotiation, while the default value is offered first. Note that the MRRU option is only offered when multi-link is enabled. The default minimum value is 68. The default maximum value is 1920. The default default value is 1500.
2	[*name*] **(pf-ppp)**[*profile*]**# mtru [min <min>] [max <max>] [default <default>]**	Configures the minimum, maximum and default value of the remote MRRU, i.e. the MRRU of the remote side that we are willing to accept. The configured range (min..max) is forced during PPP LCP negotiation, while the default value is used when not offered. The default minimum value is 68. The default maximum value is 1920. The default default value is 1500.

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

```
node(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

```
node(pf-ppp)[PPPoE]#show profile ppp PPPoE

Profiles:
---------

Name:                       default
LCP Configure-Request:      interval 3000 ms, max 10
LCP Configure-Nak:          max 5
LCP Terminate-Request:      interval 3000 ms, max 2
LCP Echo-Request:           interval 10000 ms, max 3
MTU:                        68 - 1920
MRU:                        68 - 1920
Callback:                   both
CHAP:                       allowed
PAP:                        allowed
Authentication:             interval 3000 ms, max 3
IPCP Configure-Request:     interval 3000 ms, max 10
IPCP Configure-Nak:         max 5
IPCP Terminate-Request:     interval 3000 ms, max 2
Van-Jacobson Compression:   allowed, max-slots 31
Van-Jacobson Decompression:allowed, max-slots 31

Name:                       PPPoE
LCP Configure-Request:      interval 3000 ms, max 10
LCP Configure-Nak:          max 5
LCP Terminate-Request:      interval 3000 ms, max 2
LCP Echo-Request:           interval 10000 ms, max 3
MTU:                        68 - 1492
MRU:                        68 - 1492
Callback:                   both
CHAP:                       allowed
PAP:                        allowed
Authentication:             interval 3000 ms, max 3
IPCP Configure-Request:     interval 3000 ms, max 10
IPCP Configure-Nak:         max 5
IPCP Terminate-Request:     interval 3000 ms, max 2
Van-Jacobson Compression:   allowed, max-slots 24
Van-Jacobson Decompression:allowed, max-slots 31
Van-Jacobson Decompression:allowed, max-slots 31
```

Debugging PPP

A set of commands is available to check the status of the PPP connection and the PPPoE session. Furthermore, two debug monitors help to analyze the dynamic behavior. The commands are listed in the order which you should follow in case you encounter problems with PPP. This procedure describes how to display PPP configuration information

Mode: Configure

Step	Command	Purpose
1	***node*(cfg) #show ppp links [** *level* **]**	Displays status and configuration information of the Link Control Protocol (LCP) and the authentication protocol(s) (PAP and/or CHAP). Check whether the states of the two protocols are 'Opened'. level specifies to level of details displayed (1..4, default is 1).
2	***node*(cfg) #show ppp networks [** *level* **]**	Displays status and configuration information of the Network Control Protocol(s) (NCP), in particular the IP Control Protocol (IPCP). Check whether the states of this protocol is 'Opened'. Under 'Local configuration options', you find the IP address proposed by this SmartNode and under 'Local acknowledged options', the IP address assigned by the remote peer. *level* specifies to level of details displayed (1..4, default is 1).
3	***node*(cfg) #show pppoe [** *name* **]**	Displays status, configuration information, and statistics of PPPoE in general and of the PPPoE session(s). Check whether state of the respective session is 'Opened'. *level* specifies to level of details displayed (1..4, default is 1).
4	***node*(cfg) #show port interface** *name*	Displays status and configuration information of the IP interface at which a PPP connection terminates. Check whether state of the interface is 'OPENED'. Under 'Local IP Address', you find the IP address assigned to the IP interface. If it does not correspond to the IP address assigned by the PPP remote peer, check whether the 'ipaddress' of the IP interface is set to 'unnumbered'.
5	***node*(cfg) #show port ethernet** *slot port*	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

```
node(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

```
node(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

```
node(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 a HDLC Link (Serial Port)

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
```

PPP over a HDLC Link (E1T1 Port)

Without authentication, numbered interface

```
context ip router

  interface myPPP
    ipaddress 172.17.1.1 255.255.255.252
    point-to-point
```

```
port e1t1 0 0
  port-type e1
  framing crc4
  encapsulation hdlc

  hdlc
    encapsulation ppp
    bind interface myPPP router

port e1t1 0 0
  no shutdown
```

PPP Dial-up over ISDN

The following modules in this section must be configured in order to use the PPP dial-up over ISDN feature:

- PPP Dialer (see page 331)
- Dial-up (see page 337)
- IP Link Supervision (see page 338)

PPP Dialer

The PPP Dialer is one of the modules you have to configure in order to use the PPP dial-up over ISDN feature. Also, consider the **dial-up** command (page 337) and the **check-connectivity** command (page 338) on the IP interface.

The dialer is responsible for establishing and tearing down call-signaling connections to the remote-access server. The dialer is a virtual interface of the circuit-switching (CS) context. Thus the dialer belongs to the domain of switch-circuit connectivity even if the established link finally runs the IP protocol.

To establish a connection, the dialer must be triggered by an external source, for example, a spoofing IP interface (an IP interface configured for **dial-up**). Whether this trigger is fired on the first packet sent over that IP interface or if the primary link to the destination goes down, is configured using the **dial-up** command on bound IP interface.

When the dialer receives the trigger event it tries to establish one or more connections to the configured destination. You can specify a list of destinations to be tried. Each destination configuration contains all information needed to dial and log in to a certain remote-access server (i.e. remote party number, login credentials, PPP parameters, etc.). However some of the parameters are indirectly configured using a PPP subscriber configuration entity.

The following configuration parameters are the same for all destinations:

- **Recovery Strategy:** If and when a retry is started after a failed dial attempt.
- **Destinations:** A list of destinations, each of which contains the parameters below.

The following configuration parameters can be configured for each destination separately:

- **Local and Remote Party Number:** Calling- and Called-E.164 number for the signaling connection.
- **Call Route:** A link to a call-router element, for example an ISDN interface. This route defines over which interface the dial-up call is placed. Instead of routing a call directly to an ISDN interface, you can also route the call to a hunt-group service that hunts for a free B-channel over multiple BRI interfaces.

- **Retry List:** A list of up to 8 retry timeouts that define if and when a retry is started for the same destination after a failed dial attempt.
- **Binding:** By binding a dialer destination to a PPP subscriber configuration entity, you define the PPP protocol parameters (e.g. multi-link) and credentials that are applied to the dial-up connection. Note that each dialer destination (link to a dial-up provider) may define its own credentials and PPP parameters. The PPP subscriber configuration entity also defines from which IP interface the dial-trigger is received.
- **PPP Profile:** A PPP profile can be used to fine-tune PPP protocol parameters.

Create a dialer

The following command creates a new PPP dialer.

Mode: context cs

Step	Command	Purpose
1	[*name*] **(ctx-cs)[router]# interface dialer <dialer-name>**	Creates a new dialer and enters its configuration mode.

Create outbound destinations

Follow the steps below to create an outbound destination instance. Each destination contains all information needed to dial and log into a certain remote-access server (i.e. remote party number, login credentials, PPP parameters, ect.). You can create more than one destination in order to fall back to another provider if the first provider is not accessible at the moment.

Mode: context cs/interface dialer

Step	Command	Purpose
1	[*name*] **(if-dialer)[***dialer***]#outbound PROVIDER1**	Creates an outbound destination where all dial-up and login information for a certain provider can be configured.
2	[*name*] **(outbound)[***provider***]#local-e164 <e164>**	Configures the calling-party number that shall be used to establish the call. This is the number of the local system.
3	[*name*] **(outbound)[***provider***]#remote-e164 <e164>**	Configures the called-party number that shall be used to establish the call. This is the number of the remote access server.
4	[*name*] **(outbound)[***provider***]#route call dest-interface <if-name>** [*name*] **(outbound)[***provider***]#route call dest-table <table-name>** [*name*] **(outbound)[***provider***]#route call dest-service <if-name>**	Specifies a destination interface for call establishment (basic interface routing) or a destination table or call service (advanced call routing). See Chapter 33, "CS interface configuration" on page 381 for further information.

Step	Command	Purpose
5	[*name*] **(outbound)**[*provider*]**#[no] retry <timeout1> [<timeout2> [timeout3>...]]**	Specifies how many times and after which timeouts the dialer shall redial the same provider if the previous dial-attempt failed. Up to 8 retry timeouts can be specified. The specified timeouts are relative to the previous try. So, if you specify a timeout list of **retry 5 10 10**, calls are made at time 0, 5, 15, 25. Use the no-form of the command to only dial once to the current destination. The dialer goes over to the next destination if the retry-list is executed without the link being established.
6	[*name*] **(outbound)**[*provider*]**#encapsulation ppp**	Defines the data protocol that shall run over the established signalling connection. Currently only PPP is available, which is the default.
7	[*name*] **(outbound)**[*provider*]**#bind subscriber <subscriber>**	Indirectly binds to a spoofing IP interface over a PPP subscriber. The PPP subscriber contains all PPP protocol parameters (e.g. authentication protocol, credentials, multi-link, etc.) See "Creating a PPP subscriber" on page 317 for further information.
8	[*name*] **(outbound)**[*provider*]**#use profile ppp <profile>**	Defines the PPP profile that shall be used. The PPP profile stores advanced PPP protocol parameters. See "Creating a PPP profile" on page 321 for further information.
9		Repeat Steps 1-8 for all required backup providers you want to use.

Configure recovery strategy

The recovery strategy defines whether and when new dial-attempts are made after all configured outbound destinations are tried. When receiving a dial-trigger from a spoofing IP interface, the dialer iterates over its list of outbound destinations once and tries to establish the first link. Each outbound destination may define a timeout list for internal retries. If this timeout list is executed without the link being established, the dialer goes over to the next destination. The recovery strategy is only needed if all configured outbound destinations did not manage to establish the link.

The default recovery strategy is as follows:

- If the first link could not be established because of call-signaling problems (e.g. no ISDN CONNECT received), the dialer retries the first destination again after 30 seconds.
- If the first signalling connection could be established (e.g. ISDN CONNECT received), and therefore you probably have to pay for the connection, but if the PPP negotiation failed, the dialer does not start another retry. The operator has to manually reset the dialer (executing the reset command in the context cs/interface dialer mode).

- If an additional link (when using multi-link PPP) was up and is accidentally disconnected, we retry to establish it after 5 seconds.
- If an additional link (when using multi-link PPP) could not be established because of call-signaling problems (e.g. no ISDN CONNECT received), the dialer retries the same destination again after 10 seconds.
- If an additional link (when using multi-link PPP) could be established (e.g. ISDN CONNECT received), and therefore you probably have to pay for the connection, but if the PPP negotiation failed, the dialer does not start another retry. It does not try to establish further links until the whole multi-link bundle is torn down.

Use the following commands to configure the recovery strategy:

Mode: context cs/interface dialer

Step	Command	Purpose
1	[*name*] **(if-dialer)**[*dialer*]**#[no] recovery initial-link on-signaling-failure <timeout>**	Configures the recovery timeout from a call-signaling failure of the first link of a multi-link bundle. The default value is 30 seconds.
2	[*name*] **(if-dialer)**[*dialer*]**#[no] recovery initial-link on-network-error <timeout>**	Configures the recovery timeout from a network-establishment error (e.g. a PPP negotiation problem) of the first link of a multi-link bundle. The default is not to automatically recover from this situation. The operation has to execute the reset command manually in this case.
3	[*name*] **(if-dialer)**[*dialer*]**#[no] recovery additional-link if-already-up <timeout>**	Configures the recovery timeout from a link teardown after the link was already up. The default value is 5 seconds.
4	[*name*] **(if-dialer)**[*dialer*]**#[no] recovery additional-link on-signaling-failure <timeout>**	Configures the recovery timeout from a call-signaling failure of an additional link of a multi-link bundle. The default value is 10 seconds.
5	[*name*] **(if-dialer)**[*dialer*]**#[no] recovery additional-link on-network-error <timeout>**	Configures the recovery timeout from a network-establishment error (e.g. a PPP negotiation problem) of an additional link of a multi-link bundle. The default is not to automatically recover from this situation. The bundle will drain out of links. When all links of the bundle are down, the dialer recovers from this situation without a manual intervention.

Create inbound destinations

You can also create inbound destinations. Inbound destinations can be used for dial-in connections. In that case the call-router must be configured to route certain calls (e.g. calls with information transfer capability of unrestricted-digital) to the dialer interface.

If you want two devices to be symmetrically connected, each being able to establish a dial-on-demand connection to the other, you have to create an outbound and an inbound destination with the same configuration parameters.

The local and remote E.164 number configuration on inbound destinations have different meanings than for outbound destinations. On inbound destinations these numbers are used to dispatch incoming calls to one of the configured inbound destinations.

Follow the steps below to create an inbound destination instance. Each destination contains all information needed to accept an incoming call and identify towards the remote party. You can create more than one inbound destination in order to filter inbound calls and to apply different PPP subscribers and PPP profile to call from different parties.

Mode: context cs/interface dialer

Step	Command	Purpose
1	[*name*] **(if-dialer)**[*dialer*]**#inbound PROVIDER1**	Creates an inbound destination to accept incoming calls from a certain remote party.
2	[*name*] **(inbound)**[*provider*]**#local-e164 <e164>**	The incoming call must have the configured called-party number to be accepted. Use the no-form of the command to accept calls to all numbers.
3	[*name*] **(inbound)**[*provider*]**#remote-e164 <e164>**	The incoming call must have the configured calling-party number to be accepted. Use the no-form of the command to accept calls from all numbers.
4	[*name*] **(inbound)**[*provider*]**#encapsulation ppp**	Defines the data protocol that shall run over the established signalling connection. Currently only PPP is available, which is the default.
5	[*name*] **(inbound)**[*provider*]**#bind subscriber <subscriber>**	Indirectly binds to a spoofing IP interface over a PPP subscriber. The PPP subscriber contains all PPP protocol parameters (e.g. authentication protocol, credentials, multi-link, etc.) See "Creating a PPP subscriber" on page 317 for further information.
6	[*name*] **(inbound)**[*provider*]**#use profile ppp <profile>**	Defines the PPP profile that shall be used. The PPP profile stores advanced PPP protocol parameters. See "Creating a PPP profile" on page 321 for further information.
7		Repeat steps 1-6 for all required backup providers you want to use.

Debug dialer functionality

The following commands show how to display information and events of a PPP dialer.

Mode: enable

Step	Command	Purpose
1	[*name*] **#show call-control provider <dialer> detail <detail>**	Shows configuration and state information about a certain dialer instance.

Mode: enable

Step	Command	Purpose
1	[*name*] **#debug dialer detail <detail>**	Enables logging of dialer events and state changes.
2	[*name*] **#debug ppp <type>**	Enables logging of PPP protocol events. See "Debugging PPP" on page 325 for further information.

Example – Dial-on demand feature

The following example shows a configuration snippet to configure a backup-interface that uses the dial-on demand feature to establish a backup-connection to the remote network. The remote network can be reached either over the dial-in infrastructure of provider 1 or provider 2. Both need different credentials.

```
context ip

  interface IF_PRIMARY
    ipaddress 10.1.1.2 255.255.255.0
    check-connectivity ping 10.1.1.1

  interface IF_SECONDARY
    ipaddress unnumbered
    point-to-point
    dial-up monitor interface IF_PRIMARY

  route 0.0.0.0 0.0.0.0 10.1.1.1 0
  route 0.0.0.0 0.0.0.0 IF_SECONDARY 1

subscriber ppp SUB_PROVIDER1
  dial out
  authentication chap pap
  identification outbound MY_NAME password MY_PASSWORD
  bind interface IF_SECONDARY

subscriber ppp SUB_PROVIDER2
  dial out
  authentication pap
  identification outbound USER password PWD
  bind interface IF_SECONDARY

context cs
```

```
interface dialer IF_DIALUP

  outbound 1 PROVIDER1
    local-e164 100
    remote-e164 0312345678
    route call dest-interface BRI0
    retry 5 10 10
    encapsulation ppp
    bind subscriber SUB_PROVIDER1
    use profile ppp default

  outbound 2 PROVIDER2
    local-e164 100
    remote-e164 0998887766
    route call dest-interface BRI0
    retry 5
    encapsulation ppp
    bind subscriber SUB_PROVIDER2
    use profile ppp default
```

Dial-up

Dial-up is one of the modules you have to configure in order to use the PPP dial-up over ISDN feature. Also consider the **check-connectivity** command (page 338) on the IP interface and the **interface dialer** mode (page 331) in context cs.

Dial-up brings up an IP interface even if the link is not established (spoofing). On some trigger conditions the spoofer gives the signal to dial or to drop the connection to dialer that is bound to the IP interface. There are three different trigger conditions available.

Dial-up on demand

Dial when the first packet is sent out that interface and drop after a certain time where no packet is routed through that interface.

Mode: context ip/interface

Step	Command	Purpose
1	[*name*] **(if-ip)** [*interface*]**#[no] dial-up on-demand [idle timeout <seconds>] [queue-limit <number>]**	Configure dial-up on demand. Idle timeout specifies the idle time in seconds before dropping connection (Default 300). Queue-limit specifies the max. number of packets to queue while the link is down (Default 4).

Dial-up on monitor

Dial when a monitored interface is reported down and drop when that interface is reported up again. A delay time prevents to dial or drop if the monitored interface changes the state only for a short time.

Mode: context ip/interface

Step	Command	Purpose
1	[*name*] **(if-ip)** [*interface*]**#[no] dial-up monitor interface <ip-interface> [dial delay <seconds>] [drop delay <seconds>] [queue-limit <number>]**	Configure dial-up to monitor the interface specified. Dial delay specifies the timeout before dialing (Default 10). Drop delay specifies the timeout before dropping connection (Default 10). Queue-limit specifies the max. number of packets to queue while the link is down (Default 4).

Dial-up nailed

Dial if possible, and never drop.

Mode: context ip/interface

Step	Command	Purpose
1	[*name*] **(if-ip)** [*interface*]**#[no] dial-up nailed [queue-limit <number>]**	Configure dial-up nailed. Queue-limit specifies the max. number of packets to queue while the link is down (Default 4).

Chapter 31 CS context overview

Chapter contents

Introduction 340
CS context configuration task list 341
Planning the CS configuration 341
Configuring general CS settings 343
Configuring the clock source 343
Debugging the clock source 344
Selecting PCM law compression 345
Configuring call routing 345
Creating and configuring CS interfaces 346
Specify call routing 346
Configuring dial tones 347
Configuring voice over IP parameters 347
Configuring ISDN ports 348
Configuring FXS ports 348
Configuring an H.323 VoIP connection 348
Configuring a SIP VoIP connection 348
Activating CS context configuration 349
Planning the CS context 352
Configuring general CS settings 353
Configuring call routing 353
Configuring VoIP settings 355
Configuring BRI ports 355
Configuring an H.323 VoIP connection 356
Activating the CS context configuration 356
Showing the running configuration 358

Introduction

This chapter gives an overview of the circuit-switching (CS) context and associated components, and describes the tasks involved in its configuration. It describes the steps needed configure voice connectivity, and refers to other chapters where a configuration topic is explained in more detail. Before reviewing the content in this chapter, read the configuration concepts as described in chapter 2, "Configuration concepts" on page 44.

The CS context is a high level conceptual entity that is responsible for all aspects of circuit signaling, switching, and emulation. Besides the CS context itself, the CS entity consists of the following (indicated by the shaded area enclosed by a dashed line in figure 47):

- The CS interfaces
- ISDN and FXS ports
- Tone-set profiles
- SIP and H.323 gateways
- VoIP profiles

The CS Context is enabled by default.

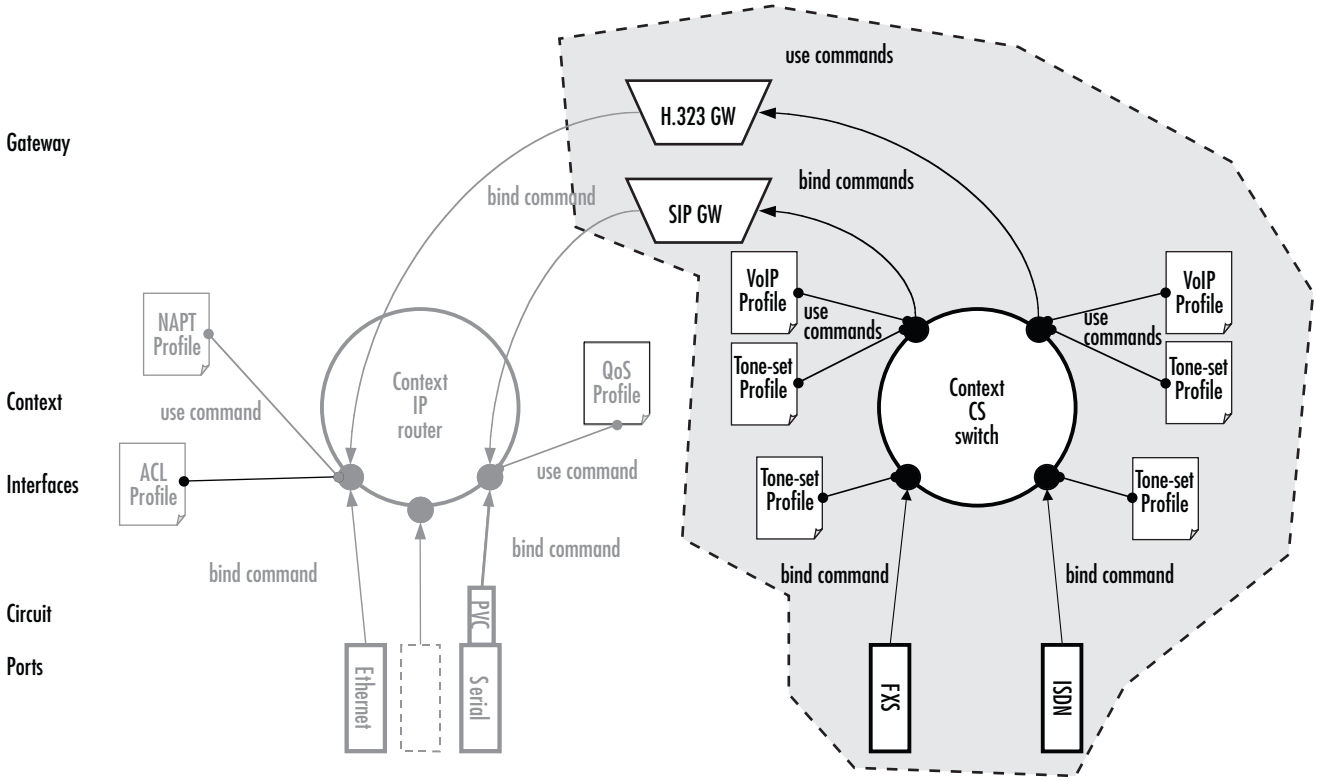


Figure 47. CS context configuration components

The CS context and its associated components route and establish voice calls. For example, the signaling for dial-up circuits is routed and the corresponding voice call circuits are switched between PSTN interfaces and via VoIP interfaces to the VoIP gateways and the IP context (see section "Configuring call routing" on page 345 for more details).

CS context configuration task list

Information needed for CS entity configuration is distributed among several configuration tasks, depending on its logical content. For example, information pertaining to call routing is described in section "Configuring call routing" on page 345. These configuration tasks can be described in other chapters; thus, to configure call routing you have to refer to chapter 33, "CS interface configuration" on page 381 and chapter 40, "Call router configuration" on page 456.

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

1. Planning the CS configuration
2. Configuring general CS settings
3. Configuring call routing
4. Configuring dial tones (advanced)
5. Configuring voice over IP settings (advanced)
6. Configuring ISDN ports
7. Configuring FXS ports
8. Configuring a H.323 VoIP connection
9. Configuring a SIP VoIP connection
10. Activating the CS context configuration

Planning the CS configuration

There are many policies and factors that can influence the CS context configuration. It depends on what your application is and how your network is configured. Several factors to consider for planning your CS configuration are listed below:

- Application/network scenario
- Peripheral devices, such as PBX or remote VoIP gateway.
- VoIP protocol
- Number and type of physical telephony ports available
- Call routing

Figure 48 shows a typical application with a remote office in an enterprise network. The example focuses on the SmartNode in the remote office. There is an ISDN phone, a personal computer, a connection to the public ISDN network, and a connection to the IP backbone. The VoIP protocol used is H.323 with a codec G.711. A call can be routed to the IP backbone and the public ISDN network depending on its prefix and number length.

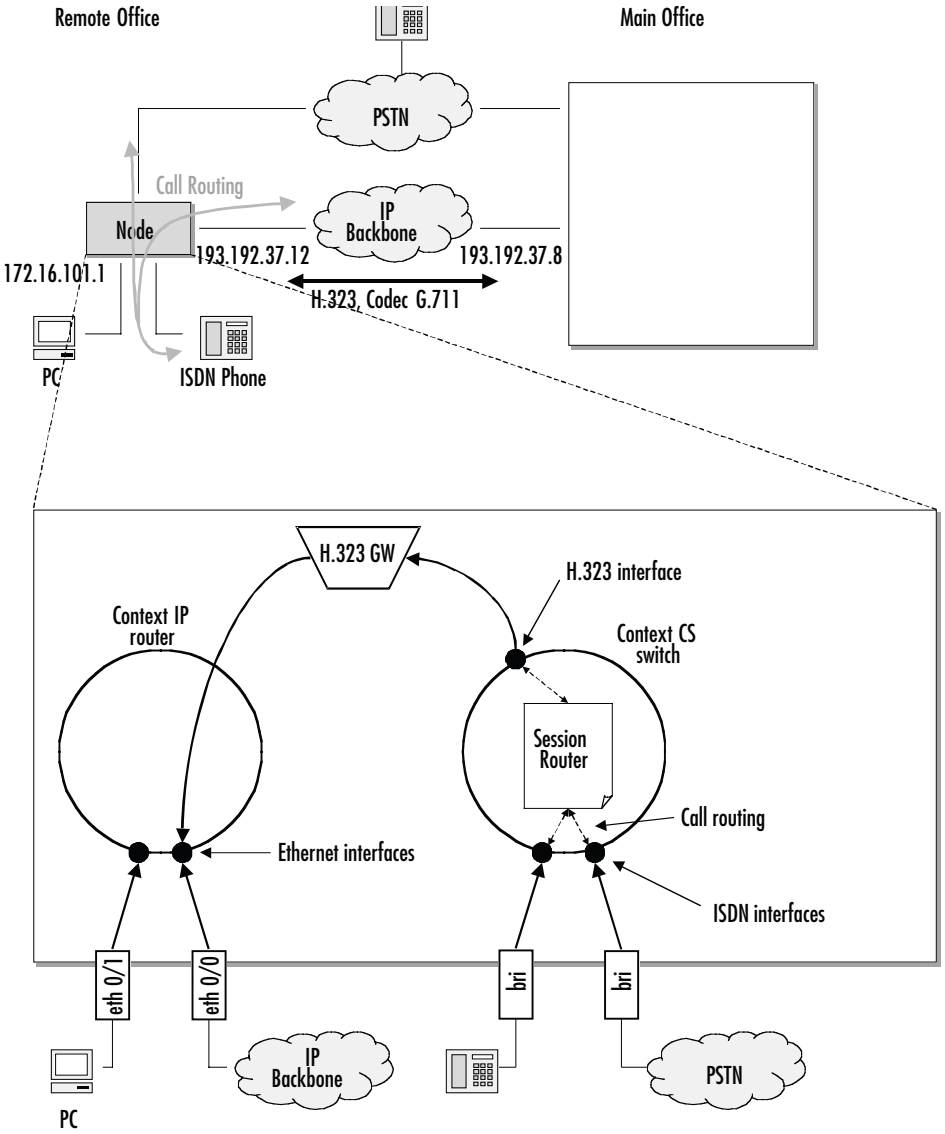


Figure 48. Remote office in an Enterprise network

An application like that shown in figure 48 would require the following CS configuration:

- Since the remote office is connected to the public switched telephone network, the clock-source comes from the corresponding ISDN port. (Described in section "Configuring general CS settings" on page 343).

Note Be careful when choosing where you get your clock source, if the clock used for packaging the ISDN voice frames is not synchronized with the remote ISDN clock, bit errors may result (such synchronization problems would probably cause a fax transmission to fail).

- Two BRI ports will be needed, the first port for the ISDN phone and the second for the public ISDN network (see section "Configuring ISDN ports" on page 348).
- Two ISDN interfaces will be needed, each bound to a BRI port (see section "Configuring call routing" on page 345)
- An H.323 interface is required in order to use H.323 (see section "Configuring call routing" on page 345)
- The call router routing tables, and the H.323 and ISDN interfaces will have to be configure to support call routing (see section "Configuring call routing" on page 345).

  Calls are routed from an ISDN phone with a number in the range of 1xx–5xx to the main office with a fallback to the PSTN. All other calls are routed from the ISDN phone to the PSTN and from the PSTN or main office to the ISDN phone.
- The H.323 gateway must be configured to use the G.711 codec (see section "Configuring an H.323 VoIP connection" on page 348)
- Two Ethernet ports and their corresponding IP interfaces will be needed.

You must not start to configure the CS context and its components until you have finished planning your voice environment. The following chapters explain how to convert the planned voice environment into the SmartWare CS configuration. The IP configuration is not a topic in this example. For more information on IP configuration refer to chapter 9, "IP context overview" on page 114.

Configuring general CS settings

There are several parameters that cannot be collected into one specific configuration task, because they are independent of the rest of the CS context configuration and apply mostly to an interface card or even to the entire SmartNode.

Configuring the clock source

A reference clock is needed for packaging the ISDN voice frames. The reference clock can be generated internally or obtained from an external source (e.g. public ISDN). SmartNode devices have a feature called 'Clock Source Hunting'. This feature allows to configure an index-based list of clock sources. The source with the lowest index has the highest priority and vice versa. On SmartNode devices populated with several PRI or BRI ports where more than one port is working in 'clock slave' mode, all these ports can be entered in the clock source list. The algorithm behind this feature always takes the first synchronized 'slave' port in the list as the current clock source. If the links of all the ports in the list are down or not synchronized, the system is falling back to its internal clock source. It is also possible to enter all PRI or BRI ports of the device in the list, independent on their clock mode. The Clock Source Hunting algorithm ignores all entered ports that are not working in 'slave' mode.

Mode: System

Step	Command	Purpose
1	***node*(sys)#clock-source** *hw-type slot port*	Add an entry to the end of the list
	***node*(sys)#clock-source** *index hw-type slot port*	Overwrite and entry at position 'index'
	***node*(sys)#clock-source before** *index hw-type slot port*	Insert an entry before position 'index'
	***node*(sys)#clock-source after** *index hw-type slot port*	Insert an entry after position 'index'
	***node*(sys)#clock-source** *index* **up** *positions*	Move entry at 'index' number of 'positions' up
	***node*(sys)#clock-source** *index* **down** *positions*	Move entry at 'index' number of 'positions' down

Debugging the clock source

To control the system behaviour at runtime, there exists a debug command with the options 'event' and 'error'. If the user enables the 'event' command, he can follow the system's Clock Source Hunting algorithm, the output informs about the occurrence of a clock source change. In addition exists a 'show' command that prints the clock source configuration and marks the ports that are synchronized. Further, the port that has been chosen by the system as the current clock source will also be displayed.

Mode: Operator execution

Step	Command	Purpose
1	***node*#[no] debug system-clock {event | error}**	Enables/Disables the system clock monitor

Mode: Operator execution

Step	Command	Purpose
1	***node*#show system-clock**	Print system clock information

```
node#show system-clock

Current clock source
====================

 t1 0 2 0

Registered clock sources
========================

 Name                        Sync
 e1 0 1 0
 t1 0 2 0                    X
 bri 0 3 0
 bri 0 3 1
 bri 0 3 2
 bri 0 3 3
 internal                    X
```

Selecting PCM law compression

The PCM law-select specifies the voice characteristic compression curve. Two values are possible: *a-Law* (used in Europe) and *µ-Law* (used in the USA).

Procedure: To set the general CS parameters

Mode: System

Step	Command	Purpose
1	***node*(sys)#clock-source internal** **or** ***node* (sys)#clock-source *slot-number port-number***	Generates the reference clock internally or specifies a specific port to receive the reference clock.
2	***node* (sys)#ic voice** *slot-number*	Changes to ic_voice mode.
3	***node* (ic-voice)[*slot-number*]#pcm law-select { aLaw | uLaw }**	Selects the PCM aLaw for Europe or uLaw for USA.

Configure: General CS settings

The following example configures the general CS parameters

```
node>enable
node#configure
node(cfg)#system
node(sys)#clock-source 1 0
node(sys)#ic voice 1
node(ic-voice)[1]#pcm law-select aLaw
node(ic-voice)[1]#exit
```

Configuring call routing

Calls through a SmartNode can be routed according to a set of routing criteria. The entity that manages call routing is called the *call router*. Calls are routed from one CS interface to another. The call router determines the destination interface for every incoming call. It supports complex call routing and call property manipulation (e.g. number manipulation) functions. See chapter 40, “Call router configuration” on page 456.

Call routing occurs in the context CS element between several CS interfaces. Accordingly, a CS context and two or more CS interfaces must be created.

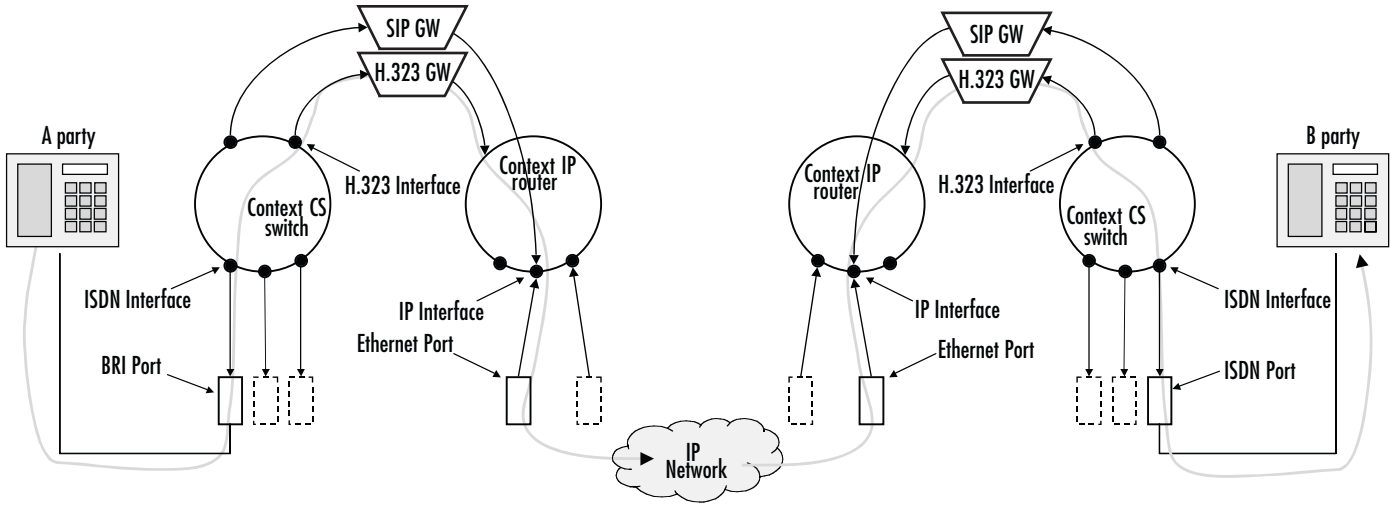


Figure 49. Direct call routing from one SmartNode to another

Figure 49 shows a call set up from the A-party on the left to the B-party on the right. The call is routed from the phone on the left-hand side over the ISDN interface directly to an H.323 interface. Once it has passed the IP context and the IP network, the other SmartNode—from the H.323 interface to the ISDN interface and then over the BRI port to the B-party phone—routes the call.

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

Creating and configuring CS interfaces

Multiple instances of CS contexts are supported. The name of the default instance is *switch*. The name and number of CS interfaces depends on your own configuration. The interfaces on the CS context represent logical connections to other equipment or networks. CS interfaces are used as source and destination in the call router. VoIP CS interfaces are bound to a gateway. Telephony ports are bound to respective interfaces.

Interface names can be any arbitrary string with a maximum of 25 characters. For ease of identification, the interface type can be a part of the name. For examples and information on how to create CS interfaces, refer to chapter 33, "CS interface configuration" on page 381.

Specify call routing

As mentioned previously, for basic call routing you can omit creating call router tables. SmartWare offers two levels of call routing:

- Basic interface routing
- Advanced call routing

Basic interface routing allows you to forward all incoming calls on a CS interface directly to a destination CS interface. The call router allows you to route calls to all available CS interfaces, based on a call property such as calling number, destination number and ISDN bearer capability and many more.

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

To configure basic interface routing refer to chapter 33, "CS interface configuration" on page 381. Other topics that belong to call routing are also explained in this chapter.

To configure advanced call routing in relation to the call router tables refer to chapter 40, "Call router configuration" on page 456. In this chapter, the differences between basic interface routing and advanced call routing are described in more detail.

Configuring dial tones

SmartWare supports country-specific, configurable, in-band dial tones that are generated for specific events, For example, alerting, and dialing or busy signals. The tones are configured in tone-set profiles that are used from a specific CS interface.

If no tone-set profile is specified, a default tone-set profile is used. In most cases, the default profile can be used, so you do not need to perform this configuration task.

Configuring voice over IP parameters

In SmartWare, there are many configurable parameters that can affect a voice over IP connection.

The *voice over IP* (VoIP) parameters are configured in the VoIP profile. A VoIP profile is used by a H.323 or SIP interface. All calls going through that interface (see figure 49 on page 346) use the settings in the VoIP profile. The following parameters are configured in the VoIP profile:

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

Refer to chapter 47, "VoIP profile configuration" on page 573 to configure general VoIP parameters. Some settings can adversely affect the voice quality perceived by the user and the bandwidth requirements of VoIP connections, so be sure you understand the meaning of the commands before changing any settings. Most of the default values of these parameters are adequate, so that you generally do not need to perform these configuration tasks.

If no VoIP profile is specified to be used on an interface, a default VoIP profile is used. In most cases, the default profile can be used, so you just need to change the default VoIP profile.

Configuring ISDN ports

BRI and E1/T1 ports represent physical ports on the SmartNode. The configuration of the ISDN ports depends on the port type (BRI, E1 or T1), and on the connected voice device. To configure the ISDN ports, refer to chapter 34, "ISDN interface configuration" on page 390.

Configuring FXS ports

FXS ports represents physical ports on the SmartNode. To configure the FXS ports, refer to chapter 43, "FXS port configuration" on page 537.

Configuring an H.323 VoIP connection

To configure a H.323 connection, you have to specify the voice codec selection used for the VoIP profile and the call signaling.

Configuring the voice codec for an H.323 connection is done on a H.323 interface by specifying the VoIP profile that shall be used. The VoIP profile contains an ordered list of codecs that must be used for codec negotiation for all calls that pass this interface. During a call setup, the first codec that is specified in the VoIP profile is taken. For information how to configure the codecs, refer to chapter 47, "VoIP profile configuration" on page 573.

H.323 offers *direct call signaling* and *gatekeeper routed call signaling* methods. For direct call signaling, you have to specify the remote terminal or gateway on each H.323 interface. Gatekeeper routed call signaling uses a gatekeeper to find the destination address. For examples and information on how to configure direct call signaling on H.323 voice connections, refer to chapter 38, "H.323 interface configuration" on page 431. To configure gatekeeper routed call signaling on H.323 voice connections, refer to chapter 45, "H.323 gateway configuration" on page 546.

Configuring a SIP VoIP connection

To configure a SIP connection, you have to specify the voice codec selection and the call signaling method for the VoIP profile.

Configuring the voice codec for a SIP connection is similar to the H.323 connection. You have to specify the VoIP profile that shall be used on a SIP interface. The VoIP profile contains an ordered list of codecs that shall be used for codec negotiation for all calls that pass this interface. During a call setup, the first codec that is specified in the VoIP profile is taken. For information on how to configure the codecs, refer to chapter 47, "VoIP profile configuration" on page 573.

You can configure the SIP gateway to register to a registrar with multiple URIs. Optionally, you can configure the SIP gateway to send all requests to an outbound proxy or redirect server.

You have several options on how to build a destination URI (To-URI) of an outgoing SIP call. You can use the called party number in conjunction with the specified domain name or you can set a specific URI by the call router, based on other call properties. For examples and information on how to configure the SIP gateway, refer to chapter 46, "Context SIP gateway overview" on page 559. To configure SIP interfaces, refer to chapter 39, "SIP interface configuration" on page 441.

Activating CS context configuration

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

In order to become functional, each interface must be bound from one port from which it receives incoming calls, and to which it forwards outgoing calls. Unlike ISDN and FXS interfaces, VoIP interfaces must be bound to a gateway.

> **Note** The difference between VoIP and PSTN interface is that VoIP interfaces are bound to a gateway while PSTN ports are bound to a CS interface. After binding to become active, the BRI, E1, T1 or FXS port must be enabled.

To bind an ISDN port to an ISDN interface, refer to chapter 34, "ISDN interface configuration" on page 390. To bind an FXS port to an FXS interface, refer to chapter 43, "FXS port configuration" on page 537. Likewise, the H.323 or SIP gateway must be enabled. Additionally, the H.323 or SIP gateway must be bound to a specific IP interface. For more information, refer to chapter 45, "H.323 gateway configuration" on page 546 or chapter 46, "Context SIP gateway overview" on page 559.

In order to become active, the CS context must be enabled. When recovering from the shutdown status, the CS context and call router configuration is checked and possible errors are indicated. The call router debug monitor can be enabled to show the loading of the CS context and call router configuration. SmartWare offers a number of possibilities to monitor and debug the CS context and call router configurations. For example, the call router debug monitor enables you to follow the sequence of tables and functions examined by the call router for each call setup. Refer to chapter 52, "VoIP debugging" on page 624 for an introduction to the configuration debugging possibilities in SmartWare.

> **Note** You can modify the configuration at runtime; changes will be active after 3 seconds. It is not necessary to shutdown the CS context before making configuration changes, a newly created or changed configuration is automatically loaded as long as the context CS is not shut down. Currently open calls are not affected by this reload.

There are several possibilities to show the actual CS context configuration. For more information on the **show** command, refer to the respective configuration chapters or to the chapter 33, "CS interface configuration" on page 381" and chapter 40, "Call router configuration" on page 456.

Procedure: Show the CS context configuration, enable the call router debug monitor and activate the CS context

Mode: Context CS

Step	Command	Purpose
1	***node*(ctx-cs)[switch]#show call-router config detail** *level*	Show the CS context configuration. *Level* could be 1..5. Level 1 shows less, level 5 shows all information.
2	***node* (ctx-cs)[switch]#debug call-router detail** *level*	Enable the call-router debug monitor. *Level* could be 1..5. Level 1 only logs errors, level 5 shows all relevant information to track calls through routing tables.
3	*node* (ctx-cs)[switch]#no shutdown	Enable the CS context, checks the interface and call router configuration
4	***node*(ctx-cs)[switch]#show call-router status detail** *level*	Show the actual state of the call router. This includes all configured tables as they were read-in from the configuration.

Example: Enable CS Context

The following example shows how to enable the call router debug monitor and how to enable the CS context. It also shows the output from the call router debug monitor.

```
node(cfg)#show call-router config detail 5
Table switch/TAB-ISDN-SERVICE:
Key              Value            Function       Dest-Type      Dest-Name
itc              -
-------------------------------------------------------------------------------
unrestricted-digital -            -              dest-interface IF-LOCAL-BA
default          -                -              dest-table     TAB-DEST-A

Table switch/TAB-DEST-A:
Key              Value            Function       Dest-Type      Dest-Name
called-e164      -
-------------------------------------------------------------------------------
0                -                MAP-CAC-ORANGE dest-interface IF-LOCAL-BA
00               -                MAP-CLI-MELON  dest-interface IF-NODE-C
07[4-6]          -                MAP-CAC-APPLE  dest-interface IF-LOCAL-BA
0336652...       -                -              dest-interface IF-NODE-B
default          -                -              dest-interface IF-LOCAL-BA

Table switch/CAC-APPLE:
Key              Value            Function       Dest-Type      Dest-Name
called-e164      called-e164
-------------------------------------------------------------------------------
(.%)             1055\1           -              -              -

...

node(cfg)#debug call-router
node(cfg)#context cs
node(ctx-cs)[switch]#no shutdown
```

```
02:14:30  CR    > Updating tables in 3 seconds...
02:14:33  CR    > [switch] Reloading tables now
02:14:33  CR    > [switch] Flushing all tables
02:14:33  CR    > [switch] Loading table 'TAB-ISDN-SERVICE'
02:14:33  CR    > [switch] Loading table 'TAB-DEST-A'
02:14:33  CR    > [switch] Loading table 'CAC-APPLE'
02:14:33  CR    > [switch] Loading table 'CAC-ORANGE'
02:14:33  CR    > [switch] Loading table 'CLI-MELON'
02:14:33  CR    > [switch] Loading table 'MAP-CAC-APPLE'
02:14:33  CR    > [switch] Loading table 'MAP-CAC-ORANGE'
02:14:33  CR    > [switch] Loading table 'MAP-CLI-MELON'
02:14:33  CR    > [switch] Loading table 'IF-LOCAL-BA-precall-service'
02:14:33  CR    > [switch] Loading table 'IF-PBX-A-precall-service'
02:14:33  CR    > [switch] Loading table 'IF-NODE-B-precall-service'
02:14:33  CR    > [switch] Loading table 'IF-NODE-C-precall-service'
node(ctx-cs)[switch]#
```

Example: Configure SmartNode in an Enterprise Network

Situation: Figure 50 shows an enterprise network with a SmartNode configured with a BRI port. A PBX, a LAN, the PSTN, and the company network are connected. The VoIP protocol used is H.323. There is no gatekeeper, so direct call signaling is used. The voice codec used is G.723, so the DTMF relay is enabled. Because no special dial tones have to be specified, the default tone-set profile is used.

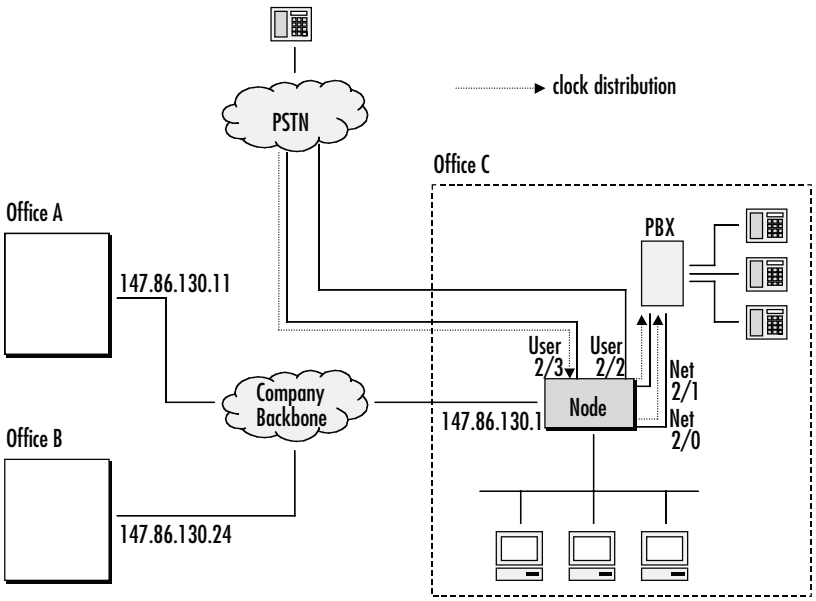


Figure 50. SmartNode in an Enterprise network

Call routing is specified as follows:

- Calls from *office C* with number *1xx* to *office A* with a fallback to PSTN
- Calls from *office C* with number *2xx* to *office B* with a fallback to PSTN
- All other calls from *office C* to PSTN

- Calls from *office A* or *B* with number *5xx* to *office C*
- All other calls from *office A* or *B* to the PSTN (local breakout)

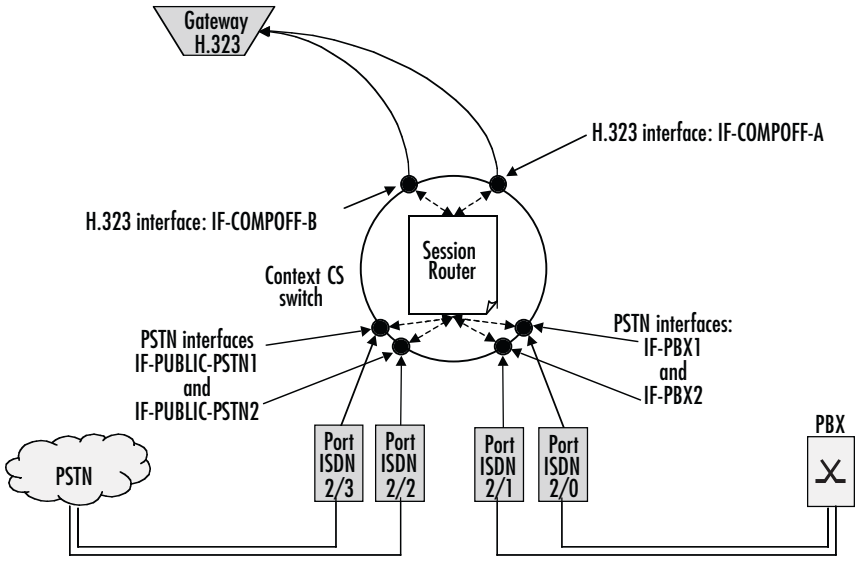


Figure 51. CS Configuration

Planning the CS context

Based on the criteria used in the previous example, the following configuration information applies (see figure 51):

- It is very important to specify from where to get the clock source for the packaging of the ISDN voice frames. In the example we are connected to the PSTN network and get the clock source from the ISDN over the ISDN port 2/3.
- We need four BRI ports, two for the PSTN and another two for the PBX. (Refer to section "Configuring ISDN ports" on page 348).
- Furthermore we need four ISDN interfaces. Then we have to bind each BRI port to one of the ISDN interfaces. A hunt group that summarizes two ISDN interfaces is configured later during call router configuration.
- For every remote H.323 device we need a H.323 interface. There are two in total. One gets the remote IP address of the SmartNode in office A, the other the IP address of the SmartNode in office B. (Refer to section "Configuring call routing" on page 345).
- We need a call router routing table to route the calls depending on the called party number. (Refer to section "Configuring call routing" on page 345).
- We further need two hunt groups, one that hunts calls to the two BRI interfaces to the PSTN and one for the two BRI interfaces to the PBX.
- Then we need two other hunt group that tries to make a call over a VoIP and if this fails, falls back to the PSTN.

- We enable DTMF relay and specify codec G.723. (Refer to section "Configuring voice over IP parameters" on page 347).
- We define H.323 direct call signaling. (Refer to section "Configuring an H.323 VoIP connection" on page 348).

Configuring general CS settings

First we set clock-source to ISDN port 2/3.

```
node>enable
node#configure
node(cfg)#system
node(sys)#clock-source 2 3
node(sys)#exit
node(cfg)#
```

Configuring call routing

Next we create the ISDN interfaces and configure call routing. Each interface is configured to route all incoming calls to the routing table TAB-CALLED-NUMBER. This table is part of the call router and configured below:

```
node(cfg)#context cs
node(ctx-cs)[switch]#interface isdn IF-PBX1
node(if-pstn)[IF-PBX1]#route call dest-table TAB-CALLED-NUMBER
node(if-pstn)[IF-PBX1]#exit

node(ctx-cs)[switch]#interface isdn IF-PBX2
node(if-pstn)[IF-PBX2]#route call dest-table TAB-CALLED-NUMBER
node(if-pstn)[IF-PBX2]#exit

node(ctx-cs)[switch]#interface isdn IF-PUBLIC-PSTN1
node(if-pstn)[IF-PUBL~]#route call dest-table TAB-CALLED-NUMBER
node(if-pstn)[IF-PUBL~]#exit

node(ctx-cs)[switch]#interface isdn IF-PUBLIC-PSTN2
node(if-pstn)[IF-PUBL~]#route call dest-table TAB-CALLED-NUMBER
node(if-pstn)[IF-PUBL~]#exit
node(ctx-cs)[switch]#
```

In addition, we create the two H.323 interfaces and configure call routing, as well as the IP address of the remote H.323 terminal, which is the IP address of the device in office A or office B, respectively.

```
node(ctx-cs)[switch]#interface h323 IF-COMPOFF-A
node(if-h323)[IF-COMP~]#route call dest-table TAB-CALLED-NUMBER
node(if-h323)[IF-COMP~]#remoteip 146.86.130.11
node(if-h323)[IF-COMP~]#bind gateway h323
node(if-h323)[IF-COMP~]#exit

node(ctx-cs)[switch]#interface h323 IF-COMPOFF-B
node(if-h323)[IF-COMP~]#route dest-table calledNumberRouting
node(if-h323)[IF-COMP~]#remoteip 146.86.130.24
node(if-h323)[IF-COMP~]#bind gateway h323
node(if-h323)[IF-COMP~]#exit
node(ctx-cs)[switch]#
```

Finally, we configure the call router. Here we create a routing table that examines the called party number of a call and routes numbers starting with a 1 and containing at least 3 digits to the hunt group that tries to reach company office A over VoIP and falls back to the PSTN. We route numbers starting with 2 and containing at least 3 digits to the hunt group that tries to reach company office B over VoIP and falls back to the PSTN. Calls with a prefix of 5 and at least 3 digits are routed to the hunt group that selects a free BRI to the PBX and all other calls are routed to the hunt group that selects a free BRI to the PSTN:

```
node(ctx-cs)[switch]#routing-table called-e164 TAB-CALLED-NUMBER
node(rt-tab)[TAB-CAL~]#route 1.. dest-service HUNT-COMPOFF-A
node(rt-tab)[TAB-CAL~]#route 2.. dest-service HUNT-COMPOFF-B
node(rt-tab)[TAB-CAL~]#route 5.. dest-service HUNT-PBX
node(rt-tab)[TAB-CAL~]#route default dest-service HUNT-PUBLIC-PSTN
node(rt-tab)[TAB-CAL~]#show call-router config
Table switch/TAB-CALLED-NUMBER:
Key              Value            Function       Dest-Type      Dest-Name
called-e164      -
-------------------------------------------------------------------------------
1..              -                -              dest-service   HUNT-COMPOFF-A
2..              -                -              dest-service   HUNT-COMPOFF-B
5..              -                -              dest-service   HUNT-PBX
default          -                -              dest-service   HUNT-PUBLIC-PSTN
node(rt-tab)[TAB-CAL~]#exit
node(ctx-cs)[switch]#
```

The hunt group HUNT-COMPOFF-A tries to reach the company office A routing the call directly to the H.323 interface IF-COMPOFF-A. When this call fails (e.g. because the data network is broken), we route the call to the PSTN hunt group. Likewise, hunt group HUNT-COMPOFF-B works, but tries to route the call to the H.323 interface IF-COMPOFF-B first.

```
node(ctx-cs)[switch]#service hunt-group HUNT-COMPOFF-A
node(rt-tab)[HUNT-CO~]#no cyclic
node(rt-tab)[HUNT-CO~]#timeout 5
node(rt-tab)[HUNT-CO~]#route call 1 dest-interface IF-COMPOFF-A
node(rt-tab)[HUNT-CO~]#route call 2 dest-service HUNT-PUBLIC-PSTN
node(rt-tab)[HUNT-CO~]#exit
node(ctx-cs)[switch]#service hunt-group HUNT-COMPOFF-B
node(rt-tab)[HUNT-CO~]#no cyclic
node(rt-tab)[HUNT-CO~]#timeout 5
node(rt-tab)[HUNT-CO~]#route call 1 dest-interface IF-COMPOFF-B
node(rt-tab)[HUNT-CO~]#route call 2 dest-service HUNT-PUBLIC-PSTN
node(rt-tab)[HUNT-CO~]#exit
node(ctx-cs)[switch]#
```

The hunt group HUNT-PBX routes the call either to the interface IF-PBX1 or IF-PBX2, depending on which interface there is a free B channel. Likewise the hunt group HUNT-PUBLIC-PSTN works on the PSTN interfaces.

```
node(ctx-cs)[switch]#service hunt-group HUNT-PBX
node(rt-tab)[HUNT-PB~]#cyclic
node(rt-tab)[HUNT-PB~]#route call 1 dest-interface IF-PBX1
node(rt-tab)[HUNT-PB~]#route call 2 dest-interface IF-PBX2
node(rt-tab)[HUNT-PB~]#exit
node(ctx-cs)[switch]#service hunt-group HUNT-PUBLIC-PSTN
```

```
node(rt-tab)[HUNT-PU~]#cyclic
node(rt-tab)[HUNT-PU~]#route call 1 dest-interface IF-PUBLIC-PSTN1
node(rt-tab)[HUNT-PU~]#route call 2 dest-interface IF-PUBLIC-PSTN2
node(rt-tab)[HUNT-PU~]#exit
node(ctx-cs)[switch]#exit
node(cfg)#
```

Configuring VoIP settings

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

```
node(cfg)#profile voip H323-VOIP-PROFILE
node(pf-voip)[H323-VO~]#codec 1 g723-6k3
node(pf-voip)[H323-VO~]#dtmf-relay
node(pf-voip)[H323-VO~]#exit
node(cfg)#
```

We want to use this profile on our H.323 interfaces:

```
node(cfg)#context cs
node(ctx-cs)[switch]#interface h323 IF-COMPOFF-A
node(if-h323)[IF-COMP~]#use profile voip H323-VOIP-PROFILE
node(if-h323)[IF-COMP~]#exit
node(ctx-cs)[switch]#interface h323 IF-COMPOFF-B
node(if-h323)[IF-COMP~]#use profile voip H323-VOIP-PROFILE
node(if-h323)[IF-COMP~]#exit
node(cfg)#
```

Configuring BRI ports

Next step is to configure the BRI ports and to bind the ports to the ISDN interfaces. We configure the layer 2 (Q.921) to use point-to-point mode and layer 3 (Q.931) for user or net operation mode:

```
node(cfg)#port bri 2 0
node(prt-bri)[2/0]#q921
node(q921)[2/0]#protocol pp
node(q921)[2/0]#q931
node(q931)[2/0]#uni-side net
node(q931)[2/0]#encapsulation cc-isdn
node(q931)[2/0]#bind interface IF-PBX1
node(q931)[2/0]#exit
node(q921)[2/0]#exit
node(prt-bri)[2/0]#no shutdown
node(cfg)#port bri 2 1
node(prt-bri)[2/1]#q921
node(q921)[2/1]#protocol pp
node(q921)[2/1]#q931
node(q931)[2/1]#uni-side net
node(q931)[2/1]#encapsulation cc-isdn
node(q931)[2/1]#bind interface IF-PBX1
node(q931)[2/1]#exit
node(q921)[2/1]#exit
node(prt-bri)[2/1]#no shutdown
node(cfg)#port bri 2 2
node(prt-bri)[2/2]#q921
node(q921)[2/2]#protocol pp
node(q921)[2/2]#q931
```

```
node(q931)[2/2]#uni-side user
node(q931)[2/2]#encapsulation cc-isdn
node(q931)[2/2]#bind interface IF-PBX1
node(q931)[2/2]#exit
node(q921)[2/2]#exit
node(prt-bri)[2/2]#no shutdown
node(cfg)#port bri 2 1
node(prt-bri)[2/3]#q921
node(q921)[2/3]#q931
node(q921)[2/3]#protocol pp
node(q931)[2/3]#uni-side user
node(q931)[2/3]#encapsulation cc-isdn
node(q931)[2/3]#bind interface IF-PBX1
node(q931)[2/3]#exit
node(q921)[2/3]#exit
node(prt-bri)[2/3]#no shutdown
```

Configuring an H.323 VoIP connection

Next we configure call signaling:

```
node(cfg)#gateway h323 h323
node(gw-h323)[h323]#no ras
node(gw-h323)[h323]#faststart
node(gw-h323)[h323]#bind interface eth0
node(gw-h323)[h323]#exit
node(cfg)#
```

Activating the CS context configuration

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

```
node(cfg)#show call-router config detail 5
Table switch/IF-PBX1-precall-service:
Key             Value           Function       Dest-Type      Dest-Name
-               -
-------------------------------------------------------------------------------
-               -               -              dest-table     TAB-CALLED-NUMBER

Table switch/IF-PBX2-precall-service:
Key             Value           Function       Dest-Type      Dest-Name
-               -
-------------------------------------------------------------------------------
-               -               -              dest-table     TAB-CALLED-NUMBER

Table switch/IF-PUBLIC-PSTN1-precall-service:
Key             Value           Function       Dest-Type      Dest-Name
-               -
-------------------------------------------------------------------------------
-               -               -              dest-table     TAB-CALLED-NUMBER

Table switch/IF-PUBLIC-PSTN2-precall-service:
Key             Value           Function       Dest-Type      Dest-Name
-               -
-------------------------------------------------------------------------------
-               -               -              dest-table     TAB-CALLED-NUMBER
```

```
Table switch/IF-COMPOFF-A-precall-service:
Key              Value            Function       Dest-Type      Dest-Name
-                -
-------------------------------------------------------------------------------
-                -                -              dest-table     TAB-CALLED-NUMBER

Table switch/IF-COMPOFF-B-precall-service:
Key              Value            Function       Dest-Type      Dest-Name
-                -
-------------------------------------------------------------------------------
-                -                -              dest-table     TAB-CALLED-NUMBER

Table switch/TAB-CALLED-NUMBER:
Key              Value            Function       Dest-Type      Dest-Name
called-e164      -
-------------------------------------------------------------------------------
1..              -                -              dest-service   HUNT-COMPOFF-A
2..              -                -              dest-service   HUNT-COMPOFF-B
5..              -                -              dest-service   HUNT-PBX
default          -                -              dest-service   HUNT-PUBLIC-PSTN
node(cfg)#
node(cfg)#show gateway h323 config detail 5
H.323 Gateway: h323
===================

 RAS Engine
 ----------

  Administrative Status:               no
  Gatekeeper-Discovery:                auto
  Gatekeepers
  Re-Registration Time:                90s
  Local Aliases
  Source Information

 Faststart:                            yes
 Early-H.245:                          no
 H.245-Tunneling:                      no
 Call-Signaling:                       147.86.130.1/1720
 Administrative Status:                close
node(cfg)#
```

Finally, activate the gateway and CS context:

```
node(cfg)#gateway h323
node(gw-h323)[gw_name]#no shutdown
node(gw-h323)[gw_name]#exit
node(cfg)#debug call-router detail 5
node(cfg)#context cs
node(ctx-cs)[switch]#no shutdown
02:30:26  CR    > Updating tables in 3 seconds...
02:30:28  CR    > [switch] Reloading tables now
02:30:28  CR    > [switch] Flushing all tables
02:30:28  CR    > [switch] Loading table 'IF-PBX1-precall-service'
02:30:28  CR    > [switch] Loading table 'IF-PBX2-precall-service'
02:30:28  CR    > [switch] Loading table 'IF-PUBLIC-PSTN1-precall-service'
```

```
02:30:28  CR    > [switch] Loading table 'IF-PUBLIC-PSTN2-precall-service'
02:30:28  CR    > [switch] Loading table 'IF-COMPOFF-A-precall-service'
02:30:28  CR    > [switch] Loading table 'IF-COMPOFF-B-precall-service'
02:30:28  CR    > [switch] Loading table 'TAB-CALLED-NUMBER'
node(ctx-cs)[switch]#
```

Showing the running configuration

The configuration script for our application looks as follows:

```
  cli version 3.00

system
  clock-source 2 3

profile voip H323-VOIP-PROFILE
  codec 1 g723-6k3 rx-length 30 tx-length 30
  codec 2 g711alaw64k rx-length 20 tx-length 20
  codec 3 g711ulaw64k rx-length 20 tx-length 20

context ip router

interface eth0
  ipaddress 147.86.130.1 255.255.225.0
  mtu 1500

interface eth1
  ipaddress 10.0.0.1 255.255.225.0
  mtu 1500

context cs switch

routing-table called-e164 TAB-CALLED-NUMBER
  route 1.. dest-service HUNT-COMPOFF-A
  route 2.. dest-service HUNT-COMPOFF-B
  route 5.. dest-service HUNT-PBX
  route default dest-service HUNT-PUBLIC-PSTN

interface h323 IF-COMPOFF-A
  bind gateway h323
  route call dest-table TAB-CALLED-NUMBER
  remoteip 146.86.130.11
  use profile voip H323-VOIP-PROFILE

interface h323 IF-COMPOFF-A
  bind gateway h323
  route call dest-table TAB-CALLED-NUMBER
  remoteip 146.86.130.24
  use profile voip H323-VOIP-PROFILE

interface isdn IF-PBX1
  route call dest-table TAB-CALLED-NUMBER

interface isdn IF-PBX2
  route call dest-table TAB-CALLED-NUMBER
```

```
interface isdn IF-PUBLIC-PSTN1
  route call dest-table TAB-CALLED-NUMBER

interface isdn IF-PUBLIC-PSTN2
  route call dest-table TAB-CALLED-NUMBER

service hunt-group HUNT-COMPOFF-A
  timeout 5
  drop-cause normal-unspecified
  drop-cause no-circuit-channel-available
  drop-cause network-out-of-order
  drop-cause temporary-failure
  drop-cause switching-equipment-congestion
  drop-cause access-info-discarded
  drop-cause circuit-channel-not-available
  drop-cause resources-unavailable
  route call 1 dest-interface IF-COMPOFF-A
  route call 2 dest-service HUNT-PUBLIC-PSTN

service hunt-group HUNT-COMPOFF-B
  timeout 5
  drop-cause normal-unspecified
  drop-cause no-circuit-channel-available
  drop-cause network-out-of-order
  drop-cause temporary-failure
  drop-cause switching-equipment-congestion
  drop-cause access-info-discarded
  drop-cause circuit-channel-not-available
  drop-cause resources-unavailable
  route call 1 dest-interface IF-COMPOFF-B
  route call 2 dest-service HUNT-PUBLIC-PSTN

service hunt-group HUNT-PBX
  cyclic
  drop-cause normal-unspecified
  drop-cause no-circuit-channel-available
  drop-cause network-out-of-order
  drop-cause temporary-failure
  drop-cause switching-equipment-congestion
  drop-cause access-info-discarded
  drop-cause circuit-channel-not-available
  drop-cause resources-unavailable
  route call 1 dest-interface IF-PBX1
  route call 2 dest-interface IF-PBX2

service hunt-group HUNT-PUBLIC-PSTN
  cyclic
  drop-cause normal-unspecified
  drop-cause no-circuit-channel-available
  drop-cause network-out-of-order
  drop-cause temporary-failure
  drop-cause switching-equipment-congestion
  drop-cause access-info-discarded
  drop-cause circuit-channel-not-available
  drop-cause resources-unavailable
```

```
  route call 1 dest-interface IF-PUBLIC-PSTN1
  route call 2 dest-interface IF-PUBLIC-PSTN2

context cs switch
  no shutdown

gateway h323 h323
  faststart
  bind interface eth0 router
  no shutdown

port ethernet 0 0
  medium 10 half
  encapsulation ip
  bind interface eth0 router
  no shutdown

port ethernet 0 1
  medium 10 half
  encapsulation ip
  bind interface eth1 router
  shutdown

port bri 2 0
  clock auto
  encapsulation q921

q921
  protocol pp
  uni-side auto
  encapsulation q931

q931
  protocol dss1
  uni-side net
  encapsulation cc-isdn
  bind interface IF-PBX1

port bri 2 0
  no shutdown

port bri 2 1
  clock auto
  encapsulation q921

q921
  protocol pp
  uni-side auto
  encapsulation q931

q931
  protocol dss1
  uni-side net
  encapsulation cc-isdn
  bind interface IF-PBX2
```

```
port bri 2 1
  no shutdown

port bri 2 2
  clock auto
  encapsulation q921

q921
  protocol pp
  uni-side auto
  encapsulation q931

q931
  protocol dss1
  uni-side user
  encapsulation cc-isdn
  bind interface IF-PUBLIC-PSTN1

port bri 2 2
  no shutdown

port bri 2 3
  clock auto
  encapsulation q921

q921
  protocol pp
  uni-side auto
  encapsulation q931

q931
  protocol dss1
  uni-side user
  encapsulation cc-isdn
  bind interface IF-PUBLIC-PSTN2

port bri 2 3
  no shutdown
```

Chapter 32 VPN configuration

Chapter contents

Introduction 363
Authentication 363
Encryption 363
Transport and tunnel modes 364
Permanent IKE Tunnels 364
Key management 364
VPN configuration task list 365
Creating an IPsec transformation profile 365
Creating an IPsec policy profile 365
Creating/modifying an outgoing ACL profile for IPsec 367
Configuration of an IP interface and the IP router for IPsec 368
Displaying IPsec configuration information 368
Debugging IPsec 369
Key management (IKE) 370
Main differences between manual & IKE IPSEC configurations 370
Creating an ISAKMP transform profile 371
Creating an ISAKMP IPSEC policy profile 372
Creating/modifying an outgoing ACL profile for IPSEC 373
Configuration of an IP interface and the IP router for IPSEC 373
Policy matching 373
Sample configuration snippet 373
Troubleshooting 374
Encrypted Voice - Performance considerations 375
Performance considerations 375
Enabling RTP encryption support 375
Using an alternate source IP address for specific destinations 376
Sample configurations 377
IPsec tunnel, DES encryption 377
SmartNode configuration 377
Cisco router configuration 378
IPsec tunnel, AES encryption at 256 bit key length, AH authentication with HMAC-SHA1-96 378
SmartNode configuration 378
Cisco router configuration 378
IPsec tunnel, 3DES encryption at 192 bit key length, ESP authentication with HMAC-MD5-96 379
SmartNode configuration 379
Cisco router configuration 379

Introduction

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

A *virtual private network* (VPN) is a private data network that uses the public telecommunications infrastructure, maintaining privacy through the use of a tunneling protocol and security procedures.

There are different technologies to implement a VPN. SmartWare applies the *internet protocol security* (IPsec) Architecture (see RFC 2401). The following sections describe the main building blocks of the IPsec architecture as implemented in SmartWare.

Authentication

Authentication verifies the integrity of data stream and ensures that it is not tampered with while in transit. It also provides confirmation about data stream origin.

Two authentication protocols are available:

- Authentication header (AH): protects the IP payload, the IP header, and the authentication header itself
- Encapsulating security payload (ESP): protects the IP payload and the ESP header and trailer, but not the IP header

Two algorithms perform the authentication:

- HMAC-MD5-96: is a combination of the *keyed-hashing for message authentication* (HMAC) and the *message digest version 5* (MD5) hash algorithm. It requires an authenticator of 128-bit length and calculates a hash of 96 bits over the packet to be protected (see RFC 2403).
- HMAC-SHA1-96: is a combination of the (HMAC) and the *secure hash algorithm version 1* (SHA1). It requires an authenticator of 160 bit length and calculates a hash of 96 bits over the packet to be protected (see RFC 2404).

Encryption

Encryption protects the data in transit from unauthorized access. Encapsulating security payload (ESP) is the protocol to transport encrypted IP packets over IP (see RFC 2406).

The following encryption algorithms are available:

	Key Length [Bit]	RFC
DES-CBC (Data Encryption Standard - Cipher Block Chaining)	56	2405
3DES-CBC (Triple Data Encryption Standard - Cipher Block Chaining)	128 or 192[a]	1851
AES-CBC (Advanced Encryption Standard - Cipher Block Chaining)	128, 192, or 256	3268

a. The 3DES algorithm uses only 112 out of the 128 Bit or 168 out of the 192 Bit as key information. Cisco only supports 192 Bit keys with 3DES.

The single DES algorithm no longer offers adequate security because of its short key length (a minimum key length 100 bits is recommended). The AES algorithm is very efficient and allows the fastest encryption. AES with a key length of 128 bits is therefore the recommended algorithm.

Transport and tunnel modes

The mode determines the payload of the ESP packet and hence the application:

- Transport mode: Encapsulates only the payload of the original IP packet, but not its header, so the IPsec peers must be at the endpoints of the communications link.
- A secure connection between two hosts is the application of the transport mode.
- Tunnel mode: Encapsulates the payload and the header of the original IP packet. The IPsec peers can be (edge) routers that are not at the endpoints of the communications link.

  A secure connection of the two (private) LANs, a 'tunnel', is the application of the tunnel mode.

Permanent IKE Tunnels

By default, IKE tunnels are established as late as possible (when the first packet is flowing through) and IKE tunnels with expired lifetimes are reestablished only in case there is traffic flowing through. With the permanent option set, IKE tunnels are established shortly after boot and are reestablished after the expiration of their lifetime even if there was no traffic flowing through.

Mode: Configure

Step	Command	Purpose
1	***node*(pf-ipsik)[*name*]#protected- network {host <local-host-ip>}|{subnet <local-subnet-address> <local-subnet-mask>}|{range <local-range-start> <local-range-end>} {host <remote-host-ip>}|{subnet <remote-subnet-address> <remote-subnet-mask>}|{range <remote-range-start><remote-range-end>} [permanent-tunnel]**	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. If the tunnel shall be established permanently the permanent-tunnel flag must be set.

Key management

The current implementation of IP 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 SmartNode and its peer.

> **Note** Depending on the processing hardware applied to *reverse engineering* a DES key, it can take from 3 hours to 3 days to break the key. Thus, for maximum security, DES keys must be manually updated regularly. AES- or 3DES-keys, because they are much more complex, take so much longer to break as to be practically infinite.

The 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 363
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.

```
node(cfg)#profile ipsec-transform AES_128
node(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 363 and "Encryption" on page 363) 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.

```
node(cfg)#profile ipsec-policy-manual ToBerne
node(pf-ipsma)[ToBerne]#use profile ipsec-transform AES_128
node(pf-ipsma)[ToBerne]#session-key inbound esp-encryption
1234567890ABCDEF1234567890ABCDEF
node(pf-ipsma)[ToBerne]#session-key outbound esp-encryption
FEDCBA0987654321FEDCBA0987654321
node(pf-ipsma)[ToBerne]#spi inbound esp 1111
node(pf-ipsma)[ToBerne]#spi outbound esp 2222
node(pf-ipsma)[ToBerne]#peer 200.200.200.1
node(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 24, "Access control list configuration" on page 253.

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.

```
node(cfg)#profile acl VPN_Out
node(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
node(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.

```
node(cfg)#context ip router
node(ctx-ip)[router]#interface WAN
node(if-ip)[WAN]#use profile acl VPN_Out out
node(if-ip)[WAN]#context ip router
node(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

```
node(cfg)#show profile ipsec-transform

IPSEC transform profiles:

Name: AES_128
 ESP Encryption: AES-CBC, Key length: 128
```

Example: Display IPsec policy profiles

```
node(cfg)#show profile ipsec-policy-manual

Manually keyed IPsec policy profiles:

Name: ToBerne, Peer: 200.200.200.1, Mode: tunnel, transform-profile: AES_128
 ESP SPI Inbound: 1111, Outbound: 2222
 ESP Encryption Key Inbound: 1234567890ABCDEF1234567890ABCDEF
 ESP Encryption Key Outbound: FEDCBA0987654321FEDCBA0987654321
```

Debugging IPsec

A debug monitor and an additional **show** command are at your disposal to debug IPsec problems.

Procedure: To debug IPsec connections

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#debug ipsec**	Enables IPsec debug monitor
2 optional	***node*(cfg)#show ipsec security-associations**	Summarizes the configuration information of all IPsec connections. If an IPsec connection does not show up, then one or more parameters are missing in the respective Policy Profile. The information 'Bytes (processed)' supports debugging because it indicates whether IPsec packets depart from ('OUT') or arrive at ('IN') the SmartNode.

Example: IPsec Debug Output

```
node(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

```
node(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)

In addition to manual keyed IPSEC connections, support for automatically keyed IPSEC connections using the Internet Key Exchange (IKE / RFC2409) protocol has been integrated, which is based on Internet Security Association and Key Management Protocol (ISAKMP / RFC2408). 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, which are required to establish one or more IPSEC security association. All the communication in QUICK MODE is protected by a previously established ISAKMP security association. Note that 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.
- In addition to the ¨profiile ipsec-transform¨, which defines the cryptographic transforms used for the IPSEC connections, it is necessary to define also a ¨profiile isakmp-transform¨, which 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 as described in Chapter 26 of the Software Configuration Guide. 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-life-time-seconds <seconds>**	Define a new maximum lifetime of the security associations in seconds.
2 (optional)	***node*(pf-ipstr)[ctx-*name*]# key-life-time-kilobytes <kilobytes>**	Define a new maximum lifetime of the security associations in kilobytes.

Creating an 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, you need to create at least one isakmp transform profile. The parameters defined in this profile are used for the negotiation of ISAKMP security associations in main mode.

The following commands can be 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*>]# authentication-algorithm md5|sha1**	Define the authentication algorithm to be used, which can be either md5 or sha1.
3	***node*(pf-ikptr)[<*name*>]# encryption des-cbc|3des-cbc|aes-cbc [key-length]**	Define the encryption and optionally the length of the encryption keys in bits to be used.
4 (optional)	***node*(pf-ikptr)[<*name*>]# key-life-time-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-life-time-sessions <sessions>**	Optionally, you can also change the default ISAKMP security association lifetime in sessions. This is the maximum number of quick modes, which can be created by the ISAKMP SA. By default there is no limit on the number of sessions.

Creating an ISAKMP IPSEC policy profile

To define all the settings and profiles needed to establish an IPSEC security association, you need to create an ISAKMP IPSEC policy profile There you can specify the ISAKMP and IPSEC transforms you created above, which should be used and other necessary parameters. You can later specify using an ACL, what traffic should be treated by a specify ISAKMP IPSEC policy.

The following commands can be 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 the specified name and enter its configuration mode.
2	***node*(pf- ipsik)[<*name*>]# authentication-method pre-shared-key <key>**	Define the pre-shared key, which sould be used to authenticate the peers. The key can be a character string of any length.
3	***node*(pf- ipsik)[<*name*>]# diffie-hellman-group {group1|group2|group5}**	Define the diffie-hellman group to be used. **Note:** The higher the group number is, the higher is the key length during the diffie-hellman exchange and the higher is 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, which is 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. Do not specify a peer, if this policy shall be used for multiple peers in transport mode. The peer can either be an IP address or a fully qualified domain name.

Step	Command	Purpose
8 (optional)	***node*(pf- ipsik)[<*name*>]# protected-network {host <local-host-ip>}|{subnet <local-subnet-address> <local-subnet-mask>}|{range <local-range-start> <local-range-end>} {host <remote-host-ip>}|{subnet <remote-subnet-address> <remote-subnet-mask>}|{range <remote-range-start> <remote-range-end>}**	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.
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 and can be done as described in Chapter 26 of the Software Configuration Guide. 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 and can be done as described in Chapter 26 of the Software Configuration Guide.

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, an FQDN might be specified as the peer instead of 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 can specify the same protection-group ID in the ISAKMP IPSEC policy profiles of all the peers, which should 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, which 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 configuration or networking problems related to IKE, the IKE module contains the following debug monitors which 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.

Encrypted Voice - Performance considerations

Firmware versions that support IKE allow encrypting and decrypting locally generated voice data streams (RTP). However, because enabling support for RTP encryption has a performance impact for the system even if RTP packets are not encrypted, this feature must be enabled manually on a per interface basis.

Performance considerations

Because encryption/decryption of RTP streams causes a very high workload on the systems CPU, this feature cannot be used on all systems without limitation. However several newer systems contain a dedicated cryptographic accelerator hardware, which does these computationally intensive tasks for the main CPU. On such systems RTP encryption has almost no impact on the overall system performance. You can see using the command 'show crypto offload', whether your systems contains the cryptographic accelerator or not.

Systems without the crEncrptedyptographic accelerator hardware will display the following line:

Crypto offload capabilities: None

Systems containing the cryptographic accelerator hardware will display the following line:

Crypto offload capabilities: DES, 3DES, AES, MD5, SHA1

On systems, which do not contain the cryptographic accelerator hardware, concurrent routing of data traffic and RTP streams through an IPSEC connection, can cause excessive jitter of the RTP packets. Therefore concurrent routing of data and RTP streams through IPSEC tunnels should be avoided on systems without the cryptographic accelerator hardware. Note that the CPU usage percentage does not give an indication about the introduced jitter, because the jitter stems form short CPU usage peaks, which are filtered out by the time averaging of the CPU workload calculation algorithm.

Enabling RTP encryption support

The following command can be used to enable/disable RTP encryption support for an IP interface. If this is enabled, RTP streams can be selected for encryption like any other data traffic using the ACL. Note that RTP encryption must be enabled on every interface, which shall be used to send or receive encrypted RTP streams.

Mode: Context ip /interface <if-name>

Step	Command	Purpose
1	***node*(if-ip)[if-*name*]# [no] rtp-encryption**	Enable or disable RTP encryption support on an IP interface.

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 like SIP and H.323.

Mode: context ip

Step	Command	Purpose
1	***node*(ctx-ip)[ctx-*name*]# [no] source-address-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 a SmartNode and a Cisco router. To interconnect two SmartNodes instead, derive the configuration for the second SmartNode by doing the following modifications:

- Swap 'inbound' and 'outbound' settings
- Adjust the 'peer' setting
- Swap the private networks in the ACL profiles
- Adjust the IP addresses of the LAN and WAN interfaces
- Adjust the route for the remote network

IPsec tunnel, DES encryption

SmartNode configuration

```
profile ipsec-transform DES
  esp-encryption des-cbc 64

profile ipsec-policy-manual VPN_DES
  use profile ipsec-transform DES
  session-key inbound esp-encryption 1234567890ABCDEF
  session-key outbound esp-encryption FEDCBA0987654321
  spi inbound esp 1111
  spi outbound esp 2222
  peer 200.200.200.1
  mode tunnel

profile acl VPN_Out
  permit ip 192.168.1.0 0.0.0.255 172.16.0.0 0.0.255.255 ipsec-policy VPN_DES
  permit ip any any

profile acl VPN_In
  permit esp any any
  permit ah any any
  permit ip 172.16.0.0 0.0.255.255 192.168.1.0 0.0.0.255
  deny ip any any

context ip router

interface LAN
  ipaddress 192.168.1.1 255.255.255.0

interface WAN
  ipaddress 200.200.200.2 255.255.255.252
  use profile acl VPN_In in
  use profile acl VPN_Out out

context ip router
  route 0.0.0.0 0.0.0.0 200.200.200.1 0
  route 172.16.0.0 255.255.0.0 WAN 0
```

Cisco router configuration

```
crypto ipsec transform-set DES esp-des
!
crypto map VPN_DES local-address FastEthernet0/1
crypto map VPN_DES 10 ipsec-manual
 set peer 200.200.200.2
 set session-key inbound esp 2222 cipher FEDCBA0987654321
 set session-key outbound esp 1111 cipher 1234567890ABCDEF
 set transform-set DES
 match address 110
!
access-list 110 permit ip 172.16.0.0 0.0.255.255 192.168.1.0 0.0.0.255
!
interface FastEthernet0/0
 ip address 172.16.1.1 255.255.0.0
!
interface FastEthernet0/1
 ip address 200.200.200.1 255.255.255.252
 crypto map VPN_DES
!
ip route 192.168.1.0 255.255.255.0 FastEthernet0/1
```

IPsec tunnel, AES encryption at 256 bit key length, AH authentication with HMAC-SHA1-96

SmartNode configuration

```
profile ipsec-transform AES_SHA1
  esp-encryption aes-cbc 256
  ah-authentication hmac-sha1-96

profile ipsec-policy-manual VPN_AES_SHA1
  use profile ipsec-transform AES_SHA1
  session-key inbound ah-authentication 1234567890ABCDEF1234567890ABCDEF12345678
  session-key outbound ah-authentication FEDCBA0987654321FEDCBA0987654321FEDCBA09
  session-key inbound esp-encryption
1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF
  session-key outbound esp-encryption
FEDCBA0987654321FEDCBA0987654321FEDCBA0987654321FEDCBA0987654321
  spi inbound ah 3333
  spi outbound ah 4444
  spi inbound esp 5555
  spi outbound esp 6666
  peer 200.200.200.1
  mode tunnel
...

Rest of the configuration, see above, just change the name of the IPsec policy pro-
file in the ACL profile ‘VPN_Out’
```

Cisco router configuration

```
crypto ipsec transform-set AES_SHA1 ah-sha-hmac esp-aes 256
!
crypto map VPN_AES_SHA1 local-address FastEthernet0/1
crypto map VPN_AES_SHA1 10 ipsec-manual
 set peer 200.200.200.2
```

```
 set session-key inbound esp 6666 cipher
FEDCBA0987654321FEDCBA0987654321FEDCBA0987654321FEDCBA0987654321
 set session-key outbound esp 5555 cipher
1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF
 set session-key inbound ah 4444 FEDCBA0987654321FEDCBA0987654321FEDCBA09
 set session-key outbound ah 3333 1234567890ABCDEF1234567890ABCDEF12345678
 set transform-set AES_SHA1
 match address 110
!
...
```

For the remainder of the configuration (see above), just change the name of the IPsec policy profile in the ACL profile *VPN_Out*

IPsec tunnel, 3DES encryption at 192 bit key length, ESP authentication with HMAC-MD5-96

SmartNode configuration

```
profile ipsec-transform TDES_MD5
  esp-encryption 3des-cbc 192
  esp-authentication hmac-md5-96

profile ipsec-policy-manual VPN_TDES_MD5
  use profile ipsec-transform TDES_MD5
  session-key inbound esp-authentication 1234567890ABCDEF1234567890ABCDEF
  session-key outbound esp-authentication FEDCBA0987654321FEDCBA0987654321
  session-key inbound esp-encryption
1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF
  session-key outbound esp-encryption
FEDCBA0987654321FEDCBA0987654321FEDCBA0987654321
  spi inbound esp 7777
  spi outbound esp 8888
  peer 200.200.200.1
  mode tunnel
...
```

For the remainder of the configuration (see above), just change the name of the IPsec policy profile in the ACL profile *VPN_Out*

Cisco router configuration

```
crypto ipsec transform-set 3DES_MD5 esp-3des esp-md5-hmac
!
crypto map VPN_3DES_MD5 local-address FastEthernet0/1
crypto map VPN_3DES_MD5 10 ipsec-manual
 set peer 200.200.200.2
 set session-key inbound esp 8888 cipher
FEDCBA0987654321FEDCBA0987654321FEDCBA0987654321 authenticator
FEDCBA0987654321FEDCBA0987654321
 set session-key outbound esp 7777 cipher
1234567890ABCDEF1234567890ABCDEF1234567890ABCDEF authenticator
1234567890ABCDEF1234567890ABCDEF
 set transform-set 3DES_MD5
 match address 110
!
```

. . .

For the remainder of the configuration (see above), just change the name of the IPsec policy profile in the ACL profile *VPN_Out*.

Chapter 33 CS interface configuration

Chapter contents

Introduction 382
CS interface configuration task list 382
Creating and configuring CS interfaces 383
Configuring call routing 384
Configuring the interface mapping tables 385
Configuring the precall service tables 388

Introduction

This chapter provides an overview of interfaces in the CS context and describes the tasks involved in their configuration. Within the CS context, an interface is a logical entity providing call signaling for incoming and outgoing calls to and from telephony ports and voice over IP gateways. It represents logical connections to other equipment or networks. CS interfaces are used as source and destination in the call router and are bound to physical ports or logical gateways.

Interface names can be any arbitrary string with a maximum of 25 characters. For ease of identification, the interface type can be a part of the name. Figure 52 illustrates the function of the CS interfaces. The types of CS interfaces are:

- PSTN interfaces telephony. Binding is done from a port to an interface.
- VoIP interface provide voice over IP settings in addition to the general CS interface parameters. These interfaces must be explicitly bound to an existing VoIP gateway.

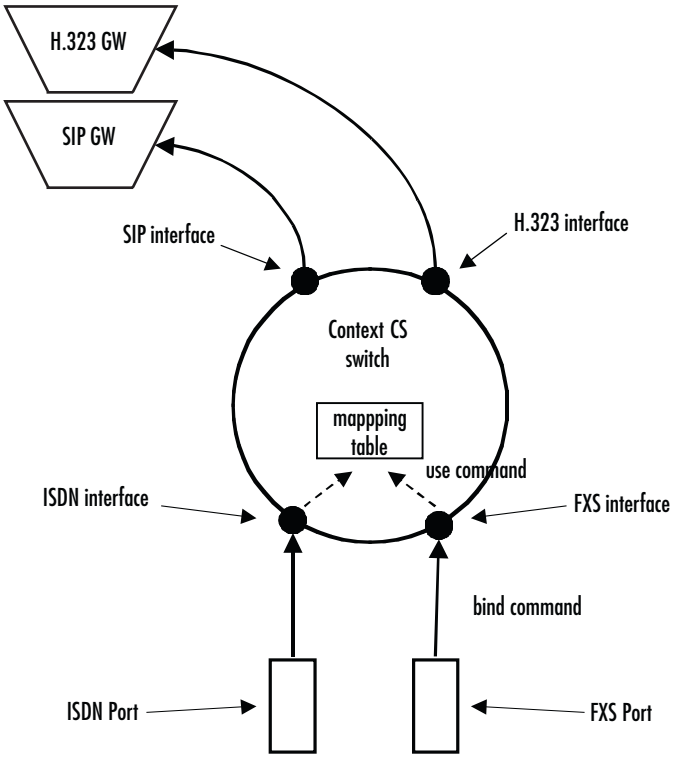


Figure 52. CS interfaces on the CS context

Interfaces can use mapping tables and precall service tables to manipulate call properties before the call is being offered to the call router.

CS interface configuration task list

Several parameters depend upon the interface type. If it is not specifically stated otherwise, the configuration task is valid for all interfaces. This is not described in this chapter, but in chapter 42, "Tone configuration" on

page 529 and chapter 47, "VoIP profile configuration" on page 573. To create and configure CS interfaces you have to perform the configuration tasks listed below.

- Creating and configuring CS interfaces
- Configuring call routing
- Configuring the interface mapping tables (optional)
- Configuring the precall service tables (optional)
- Configuring interface type specific parameters

Creating and configuring CS interfaces

To configure CS interfaces, you must first enter the CS context mode where you can create and configure your required interface through the CS interface configuration mode. Each interface has a name that can be any arbitrary string of not more than 25 characters. Use a name describing the purpose of the interface, as shown in the examples or—for ease of identification—the interface type can be used as part of the name. Already-defined CS interfaces can be displayed or deleted as described in the following table.

Procedure: Create and configure CS interfaces.

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#context cs**	Enter the CS Context Configuration Mode.
2	***node*(ctx-cs)[switch]#interface** *if-type if-name*	Enter the CS Interface Configuration Mode & select the CS interface with type *if-type* and name *if-name* for configuration. Valid interface types are **h323**, **sip**, **isdn** and **fxs**.
3	***node*(if-*type*)[*if-name*]#...**	Perform the configuration tasks to configure the CS interface.
4	***node*(ctx-cs)[*switch*]#show call-control provider**	Display the configuration of the current CS interface.
5	***node*(if-*type*)[*if-name*]#exit**	Go back to the CS Context Configuration Mode
6		Repeat step 1 to 5 to create and configure your CS interfaces
7	***node*(ctx-cs)[*switch*]#show call-control provider** **or** ***node*(ctx-cs)[*switch*]#show call-control status**	Display already defined CS interfaces. **Note:** The show call-control provider command can also be used to display the configuration details of a provider either by specifying its name as a parameter or by being inside its configuration mode.
8	***node*(ctx-cs)[*switch*]#no interface** *if-type if-name*	Delete an existing interface.

Examples: Create CS interfaces and delete another

The following example shows how to create and configure an interface, how to display it, and how to delete another.

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface isdn IF-PBX1
node(if-pstn)[IF-PBX1]#route call dest-interface TAB-CALLED-NUMBER
node(if-pstn)[IF-PBX1]#show call-control provider
Provider: IF-PBX1
=================

 Binding:                              (none)
 Protocol:                             (unknown)
 DTMF Dialing:                         disabled
 Tone-Set Profile:                     (none)
 PSTN Profile:                         default
 Routing Destination:                  router (IF-PBX1-precall-service)
 Active Endpoints:                     0
 Suspended endpoints:                  0
node(if-pstn)[IF-PBX1]#exit
node(ctx-cs)[switch]#show call-control provider
Call Control: switch
====================

 Providers
 ---------

  local
  router
  sn43
  IF-PBX1
  IF-PBX2
  IF-PUBLIC-PSTN1
  IF-PUBLIC-PSTN2
  IF-COMPOFF-A
  HUNT-COMPOFF-A
  HUNT-PBX
  HUNT-PUBLIC-PSTN
node(ctx-cs)[switch]#no interface isdn IF-PBX1
node(ctx-cs)[switch]#
```

Configuring call routing

SmartWare offers two levels of call routing: basic interface routing and advanced call routing. Basic interface routing allows you to forward all incoming calls on a CS interface to a destination CS interface.

Advanced call routing allows you to route calls to all available CS interfaces, based on a criteria such as calling number, destination number, ISDN bearer capability, or other call properties. Using mapping tables, you can modify call properties like the calling or called party number, URI, etc. Furthermore, you can collect numbers using the digit-collection feature of called party number routing tables. Call services like hunt or distribution groups can be used to distribute calls to multiple destination interfaces.

In the environment of the CS interfaces, it is necessary to specify whether the call will be routed directly to another CS interface (basic interface routing) or to a first lookup table from the call router (advanced call routing).

In this chapter. only the configuration task on a CS interface is described. For configuration of the call routing tables, mapping tables and call services refer to chapter 40, "Call router configuration" on page 456, which also describes the difference between the two levels of call routing in more detail.

Procedure: To configure basic interface routing

Mode: Context CS

Step	Command	Purpose
1	***node*(ctx-cs)[switch]#interface** *if-type if-name*	Enters CS Interface Configuration Mode and configure interface *if-type* with name *if-name*
2	***node*(if-*type*)[*if-name*]#route call dest-interface** *if-name* *or* ***node*(if-*type*)[*if-name*]#route call dest-table** *table-name* *or* ***node*(if-*type*)[*if-name*]#route call dest-service** *service-name*	Specifies a destination interface for incoming calls (basic interface routing) or a destination table or call service (advanced call routing)
3	***node*(*if- type*)[*if-name*]#exit**	Returns to CS context configuration mode
4		Repeat steps 1–3 for all the required CS interfaces

Example: Configure call routing

The following example shows how to configure basic interface routing.

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface pstn IF-PBX1
node(if-pstn)[IF-PBX1]#route call dest-interface IF-H323-0
node(if-pstn)[IF-PBX1]#exit
node(ctx-cs)[switch]#
```

Configuring the interface mapping tables

Call router mapping tables are normally used by the call router to manipulate call properties during the call setup phase, i.e. when a call arrives on a CS interface and is routed to another interface through routing and mapping tables. This imposes a limitation to call property manipulation: When a call property like a party's number is changed during a call, the call is not routed through the call router again and thus, the mapping tables are not processed for the new number. Call property manipulation, e.g. removing a prefix from a number, cannot be done for the new number.

Consider, for example, an ISDN call, which may send a connected party number in the Connect message. This connected party number has the same meaning as the original called party number, but may differ from it. Another example of a call property that changes during a call is a SIP call transfer. A SIP call may be transferred

to another user agent having a different URI than the called one. This new URI as well as the derived E.164 number cannot be manipulated using the call router before presenting it to the other party.

To circumvent this limitation, you can use mapping tables directly on an interface. In that case the mapping tables can be thought as input or output filters, which manipulate call properties at any stage of a call.

As with the SIP transfer example, differentiating called from calling party properties does not make sense for these manipulations, because the calling as well as the called party can be transferred in a SIP call. Therefore, mapping tables that are used on an interface manipulate both at the same time, called and calling party properties!

You can chose different mapping tables for filtering parameters in each direction, input and output. While an input mapping table is applied to all properties that are received by the port or gateway that is bound to the interface before sending them to the peer interface in the CS context, an output mapping table is applied to all properties before sending them to the bound port or gateway.

Refer to the chapter 40, "Call router configuration" on page 456 for more information about how to create and configure mapping tables.

Procedure: To use mapping tables to filter properties on an CS interface

Mode: Context CS

Step	Command	Purpose
1	***node*(ctx-cs)[switch]#interface** *if-type if-name*	Enters CS Interface Configuration Mode and configure interface *if-type* with name *if-name*
2	***node*(if-*type*)[*if-name*]#use mapping-table in** *table-name* *and/or* ***node*(if-*type*)[*if-name*]#use mapping-table out** *table-name*	Specifies an input and/or an output mapping table that shall be applied to all call properties in the specified direction.

Example: Use interface mapping tables for dialing plan conversion

The following example shows how to configure a dialing plan conversion on an interface. In this case, you can plan your call-routing tables to deal only with international numbers while converting private numbers on the CS interface that interfaces the private network.

```
node(ctx-cs)[switch]#mapping-table e164 to e164 PRIV-TO-GLOB
node(map-tab)[PRIV-TO~]#map (..) to 00419988825\1
node(map-tab)[PRIV-TO~]#exit
node(ctx-cs)[switch]#mapping-table e164 to e164 GLOB-TO-PRIV
node(map-tab)[GLOB-TO~]#map 00419988825(..) to \1
node(map-tab)[GLOB-TO~]#exit
node(ctx-cs)[switch]#interface isdn IF-PHONES
node(if-isdn)[IF-PHON~]#route dest-table TAB-CALLED-NUMBER
node(if-isdn)[IF-PHON~]#use mapping-table in PRIV-TO-GLOB
node(if-isdn)[IF-PHON~]#use mapping-table out GLOB-TO-PRIV
node(if-isdn)[IF-PHON~]#exit
node(ctx-cs)[switch]#
```

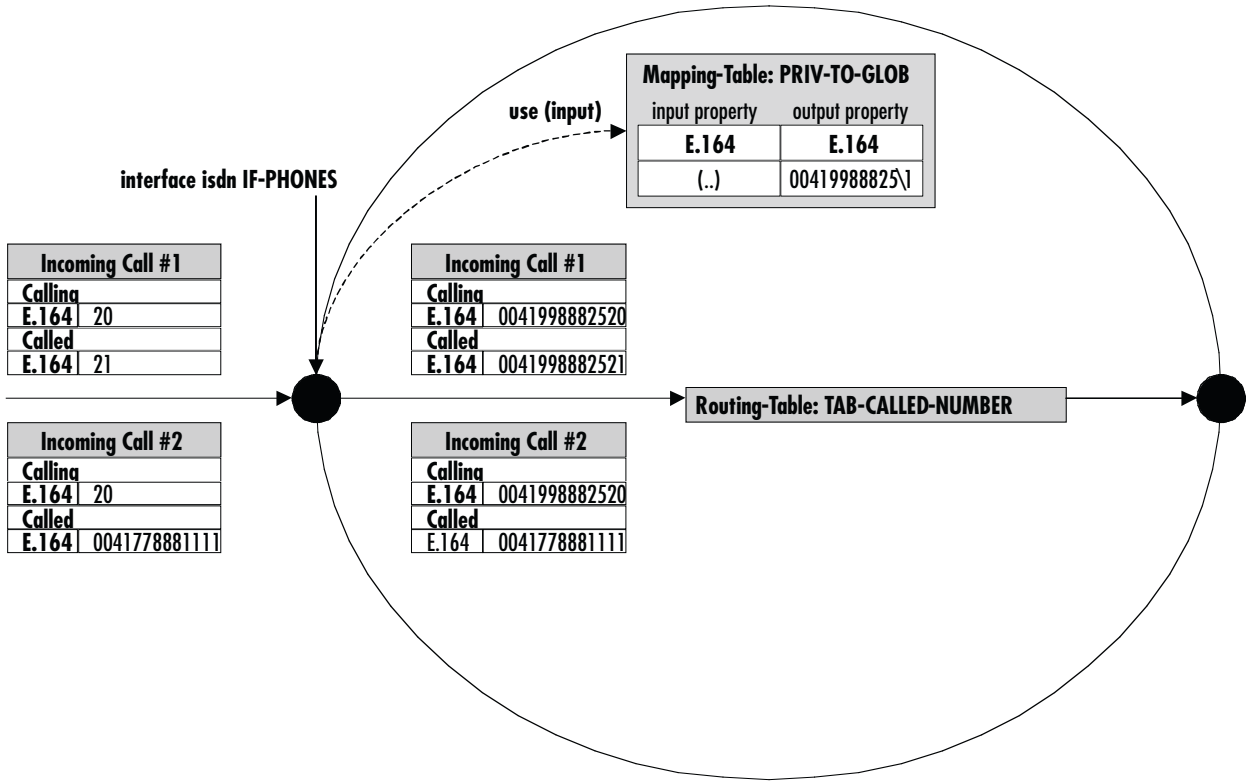


Figure 53. Incoming call passing an interface mapping table

Figure 53 shows two incoming calls arriving to the ISDN interface IF-PHONES. The calling and called party numbers are private numbers containing only two digits. Before accessing the call router, those numbers can be transformed into the global numbering plan. Which is why the interface was configured to use mapping table PRIV-TO-GLOB on all incoming call properties.

Incoming call #1 originally has a calling party number of 20 and a called party number of 21. Before offering this call to the call router, mapping table PRIV-TO-GLOB is applied to the called party number and the calling party number. The mapping table adds a prefix of 00419988825 to the called and calling party number.

Incoming call #2 originally has a calling party number of 20 but already a called party number of the global numbering plan. Again, the mapping table is applied to both number, but only the calling party number of 20 is translated into 0041998882520. The called party number does not match an entry in the mapping table, so it is not changed.

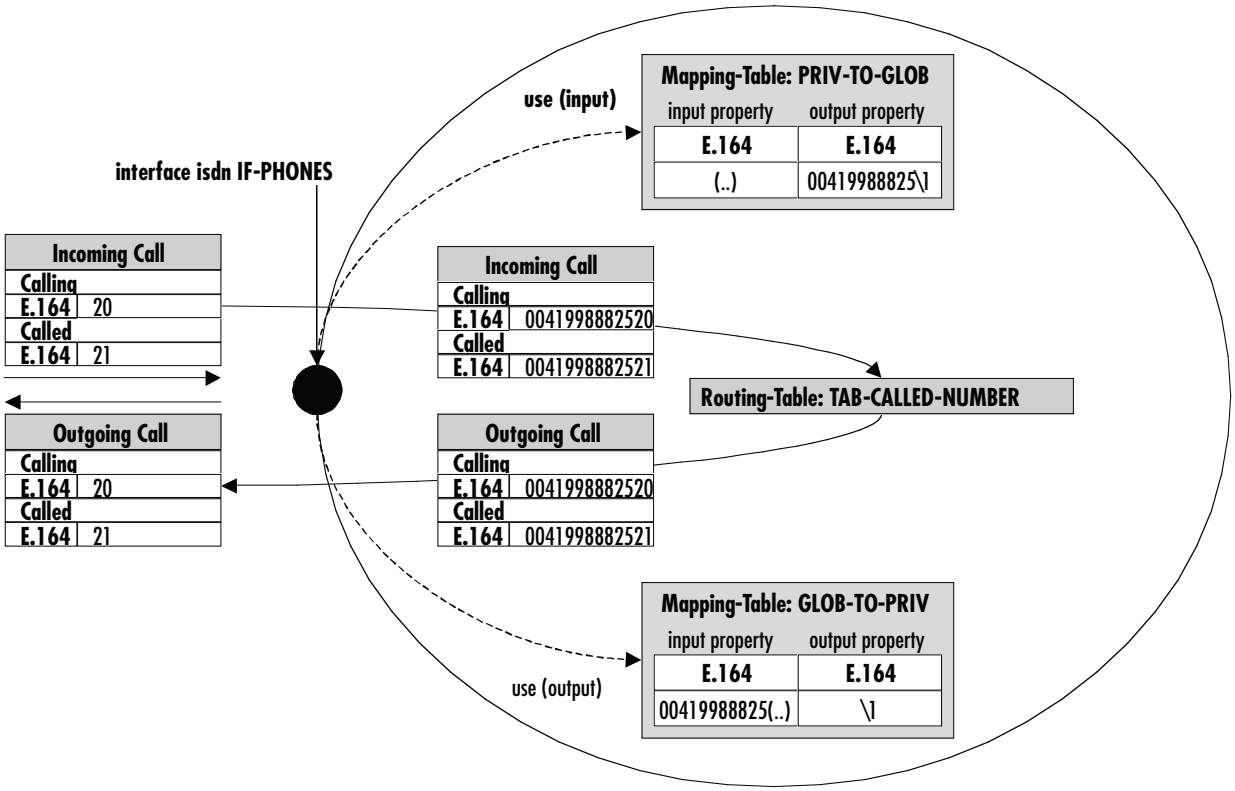


Figure 54. Call passing an input and an output mapping table

Let's assume we manipulate an incoming ISDN call using the PRIV-TO-GLOB mapping table as in the previous example. Figure 54 shows this situation again. Let's further assume the call router routes back the call to the interface IF-PHONES. In that case, the output mapping table used on this interface is applied to all call parameters. The calling and called party number is transformed form the global to the private numbering plan before the call is offered to the remote ISDN terminal.

Note For interface mapping you can use only mapping tables that examine general call parameters. For example, you cannot use a called-e164 to called-e164 mapping table, use a e164 to e164 mapping table instead.

Configuring the precall service tables

Precall service mapping tables are used to convert dialed special numbers like *61 to invocation commands for supplementary services like call-waiting, etc. Precall service tables are configured as part of the call router in the context CS configuration mode. Precall service tables are used on an FXS interface where the attached phone should be able to activate or deactivate services by dialing a special number. SmartWare currently supports the following service commands:

- **activate-cw**—Activates call-waiting on the interface that uses the precall service table. Once activated a second incoming call is possible on the interface. The second call is announced to the first call. The user can then decide whether to accept or reject the new call.
- **deactivate-cw**—Deactivates call-waiting on the interface that uses the precall service table.

- **interrogate-cw**—Detects whether or not the call-waiting supplementary service is active on the interface that uses the precall service table.

Note Currently you can only use precall service tables on FXS interfaces.

Procedure: To create precall service table and use it on an FXS interface

Mode: Context CS

Step	Command	Purpose
1	***node*(ctx-cs)[switch]#precall-service-table** *table-name*	Creates a new table that maps special numbers into supplementary service invocation commands
2	***node*(pcs-tab)[*table-name*]#map** *special-number* **to** *command*	Adds a new entry to map a *special-number* into a supplementary service invocation *command*.
3		Repeat Step 2 to add other special number mappings.
4	***node*(pcs-tab)[*table-name*]#exit**	Returns to context CS Configuration Mode
5	***node*(ctx-cs)[switch]#interface fxs** *if-name*	Enters FXS Interface Configuration Mode of interface *if-name*
6	***node*(if-fxs)[*if-name*]#use mapping-table precall-service** *table-name*	Uses the precall service table created with step 1 to 4 on this FXS interface.

Example: Create and use a precall service table

The following example shows how to create a precall service table that treats *43# as activation command for the call-waiting supplementary service, while #43# is used to deactivate call-waiting and *#43# is used to query the call-waiting supplementary service:

```
node(ctx-cs)[switch]#precall-service-table SUPP-SVC
node(pcs-tab)[SUPP-SVC]#map *43# to activate-cw
node(pcs-tab)[SUPP-SVC]#map #43# to deactivate-cw
node(pcs-tab)[SUPP-SVC]#map *#43# to interrogate-cw
node(pcs-tab)[SUPP-SVC]#exit
node(ctx-cs)[switch]#interface fxs IF-PHONE
node(if-fxs)[IF-PHONE]#use mapping-table precall-service SUPP-SVC
node(if-fxs)[IF-PHONE]#exit
node(ctx-cs)[switch]#
```

Chapter 34 ISDN interface configuration

Chapter contents

Introduction 391
ISDN interface configuration task list 391
Configuring DTMF dialing (optional) 392
Configuring an alternate PSTN profile (optional) 392
Configuring ringback tone on ISDN user-side interfaces 393
Configuring call waiting (optional) 393
Disabling call-waiting on ISDN DSS1 network interfaces 393
Configuring Call-Hold on ISDN interfaces 394
Enabling Display Information Elements on ISDN Ports 394
Configuring date/time publishing to terminals (optional) 394
Sending the connected party number (COLP) (optional) 395
Enabling sending of date and time on ISDN DSS1 network interfaces 395
Defining the 'network-type' in ISDN interfaces 395
ISDN Explicit Call Transfer support (& SIP REFER Transmission) 395
ISDN Advice of Charge support 397
ISDN DivertingLegInformation2 Facility 401
Transmit Direction 401
Receive Direction 401
T1 Caller-Name Support 401

Introduction

This chapter provides an overview of ISDN interfaces, and the tasks involved in their configuration. This chapter does not explain the basic configuration steps equal to all CS interfaces. Information about basic interface configuration can be found in the general chapter about CS interface configuration (see chapter 33, "CS interface configuration" on page 381)

An ISDN interface represents the connection of an ISDN signaling channel to the call control. It encapsulates the ISDN layer 3 protocol of an ISDN port's D-channel, allows incoming and outgoing calls on this port, controls its B-channels and provides a set of services.

There is a one-to-one relation between the port and the interface: Only one port can bind to an existing interface, and there must be a port that binds to the interface for the interface to become functional (see figure 55).

An ISDN interface can encapsulate user and network side of the following protocols: DSS1, NI2, NTT. The settings are automatically taken from the port that binds to the interface, and changes on the port are automatically reflected on the interface.

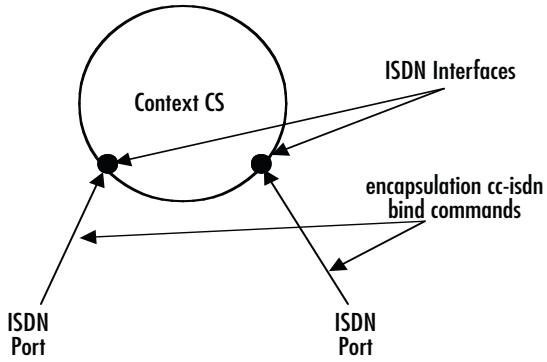


Figure 55. ISDN interfaces on the CS context

ISDN interface configuration task list

This section describes the configuration tasks for ISDN interfaces. There are no mandatory configurations on ISDN interfaces, because all protocol relevant settings are inherited from the port that binds to the interface.

The settings on the interface are those of basic CS interface configuration, as well as settings for interoperability and supplementary services:

- Configuring ringback tone on ISDN user-side interfaces
- Configuring call waiting (optional)
- Disabling call waiting on ISDN DSS1 network interfaces
- Configuring date/time publishing to terminals (optional)
- Enabling sending of date and time on ISDN DSS1 network interfaces
- Defining the 'network-type' in ISDN interfaces
- ISDN explicit call transfer, SIP REFER transmission
- ISDN Advice of Charge support

Configuring DTMF dialing (optional)

Most ISDN terminals support two modes of call setup: En-bloc dialing and overlap dialing. En-bloc dialing transports the full called party information in the first SETUP message from the terminal. This means that the user must dial the number before going off-hook. Overlap dialing transports the called-party number digit by digit, after the first SETUP message, which contains no called-party information at all. Combinations between en-bloc and overlap dialing are possible.

Most terminals use ISDN *keypad facility* messages to transport digits one-by-one in overlap dialing. But some terminals, especially terminal adapters for analog devices, might transport the digits only using DTMF tones, without associated keypad facility messages.

The **DTMF dialing** command enables the ISDN port for the use with such devices.

Be sure to only use this command when needed. Otherwise, called party information can be corrupted because the digits arrive twice, as keypad facility messages and also as DTMF tones.

Procedure: To enable DTMF dialing

Mode: Interface ISDN

Step	Command	Purpose
1	***node*(if-isdn)[*if-name*]#[no] dtmf-dialing**	Enables/Disables DTMF dialing (default: disabled)

Example: Enable DTMF dialing

The following example shows how to enable DTMF dialing for a given ISDN interface.

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface isdn MyIsdnIf
node(if-isdn)[myIsdnIf]#dtmf-dialing
```

Configuring an alternate PSTN profile (optional)

The PSTN profile contains the configuration for data/voice transmission on circuit-switched channels (see chapter 48, "PSTN profile configuration" on page 597). In the case of ISDN interfaces, the PSTN profile applies to the ISDN B-Channels associated with the interface.

There is a PSTN profile named *default*, which always exists in the system. If no different PSTN profile name is explicitly configured on the ISDN interface, the profile *default* is used.

Procedure: To define an alternate PSTN profile for the ISDN interface

Mode: Interface ISDN

Step	Command	Purpose
1	***node*(if-isdn)[*if-name*]#[no] use profile pstn** *profile-name*	Defines an alternate PSTN profile to be used for this ISDN interface/Reverts the setting to its default (use profile PSTN *default*)

Example: Configure an alternate PSTN profile

The following example shows how to replace the PSTN profile *default* of the ISDN interface with the PSTN profile *myprofile*.

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface isdn myIsdnIf
node(if-isdn)[myIsdnIf]#use profile pstn myprofile
```

Configuring ringback tone on ISDN user-side interfaces

If a ring-back tone needs to be played towards the PSTN from an ISDN user-side interface, this can be forced using the following command.

Mode: interface isdn <if-name>

Step	Command	Purpose
1	**[***name***] (pf-isdn)[***if-name***]# [no] user-side-ringback-tone**	Enables ringback tone to be played on ISDN user-side interfaces. Default: *disabled*.

Configuring call waiting (optional)

The term "call waiting" is used as follows in this context: If the port bound to this interface is configured to be network side, and both ISDN B-Channels are engaged with calls, and there is a new outgoing call over this interface, the interface can

a) Signal the new call to all connected terminals, although both B-Channels are in use. One terminal can then put its current call on hold to accept the new one (putting the call on hold frees its B-Channel).

b) Not signal the new call, because there is no B-Channel available. This is the desired behavior particularly if the bound port is part of a hunt-group, and no user terminals are connected.

Default behavior is a), using the command below in the inverted form, behavior b) is selected.

Procedure: To configure call waiting

Mode: Interface ISDN

Step	Command	Purpose
1	***node*(if-isdn)[***if-name***]#[no] call-waiting**	Enable/disable call waiting feature as described above (default: enabled)

Disabling call-waiting on ISDN DSS1 network interfaces

This procedure disables support for call-waiting on an ISDN DSS1 network interface.

Mode: interface isdn <if-name>

Step	Command	Purpose
1	**[***name***] (if-isdn)[***if-name***]# no call-waiting**	Disable call-waiting.

Configuring Call-Hold on ISDN interfaces

Normally, the call-hold feature is disabled on ISDN point-to-point links and enabled on ISDN point-to-multipoint links. However, you can manually enable or disable the Call-Hold feature using the following command: The default setting can be achieved using the 'auto' configuration option.

Mode: interface isdn

Step	Command	Purpose
1	***node*(if-isdn)[if-*name*]# call-hold {auto|enable|disable}**	Enable or disable the call-hold functionality for an isdn interface. If 'auto' is selected, call-hold is automatically disabled on p2p links and enabled on p2mp links. Default: auto

Enabling Display Information Elements on ISDN Ports

By default no display information elements are sent in ISDN signaling messages. You can enable sending of ISDN Display Information elements in ISDN signaling messages using the following command.

Mode: interface isdn

Step	Command	Purpose
1	***node*(if-isdn)[if-*name*]# [no] display emit**	Enable sending of the display information element for an isdn interface. Default: disabled

Configuring date/time publishing to terminals (optional)

ISDN allows to propagate current time and date information from a port configured as network to the connected terminals. You can configure each ISDN interface to propagate the current SmartNode system time and date to the connected terminals with the following command:

Procedure: To configure date and time publishing

Mode: Interface ISDN

Step	Command	Purpose
1	***node*(if-isdn)[*if-name*]#[no] isdn-date-time**	Enable/disable publishing of system time to connected ISDN terminals (default: disabled)

Date and time information can only be contained in the ISDN CONNECT message. This message is only delivered to a terminal when a call from the terminal to the SmartNode is made, and reaches connected state.

Sending the connected party number (COLP) (optional)

Sending the connected party number (COLP) can be suppressed by the command *send-connected- party-number*.

Mode: context cs/interface isdn

Step	Command	Purpose
1	***node*(if-isdn)[*if-name*]#[no] send-connected-party-number**	Enables/Disables sending the connected-party-number. Default: enabled.

Enabling sending of date and time on ISDN DSS1 network interfaces

This procedure enables sending of date and time information on an ISDN network side interface.

Mode: interface isdn <if-name>

Step	Command	Purpose
1	**[*name*] (if-isdn)[*if-name*]# isdn-date-time**	Enable sending of date and time.

Defining the 'network-type' in ISDN interfaces

The following command defines the location code to be inserted in ISDN causes code information elements.

Mode: interface isdn <if-isdn>

Step	Command	Purpose
1	**[*name*] (if-isdn)[*if-name*]# network-type [international|private|public|transit|user}**	Defines the type of network to which the system belongs.

ISDN Explicit Call Transfer support (& SIP REFER Transmission)

Additional call transfer support is enabled by default for ISDN interfaces (BRI ports) by accepting or rejecting explicit call-transfer (ECT) invocations. An ISDN phone that is connected to a BRI port and that has two active calls can send an ECT invocation to connect the two calls inside the device. An ISDN interface can be configured to accept or reject ECT invocations.

SmartWare detects calls that are looped internally, i.e. calls that leave the device over the same ISDN interface over which they enter the device. If an internal loop is detected for an ISDN interface bound by an ISDN user port, SmartWare sends an explicit call-transfer (ECT) to push back the call to the connected network as soon as the call is connected. An ISDN interface can be configured to emit ECT invocations.

SIP interfaces react similarly to internally looped calls. If a call leaves the device over the same SIP gateway over which it entered the device, SmartWare sends a REFER message to one of the remote user agents to transfer the call to the two parties. A SIP interface can be configured to emit REFER messages.

Figure 56 shows an example scenario where a SIP network connects two devices to give a home office (HO) access to a PBX in the central office (CO).

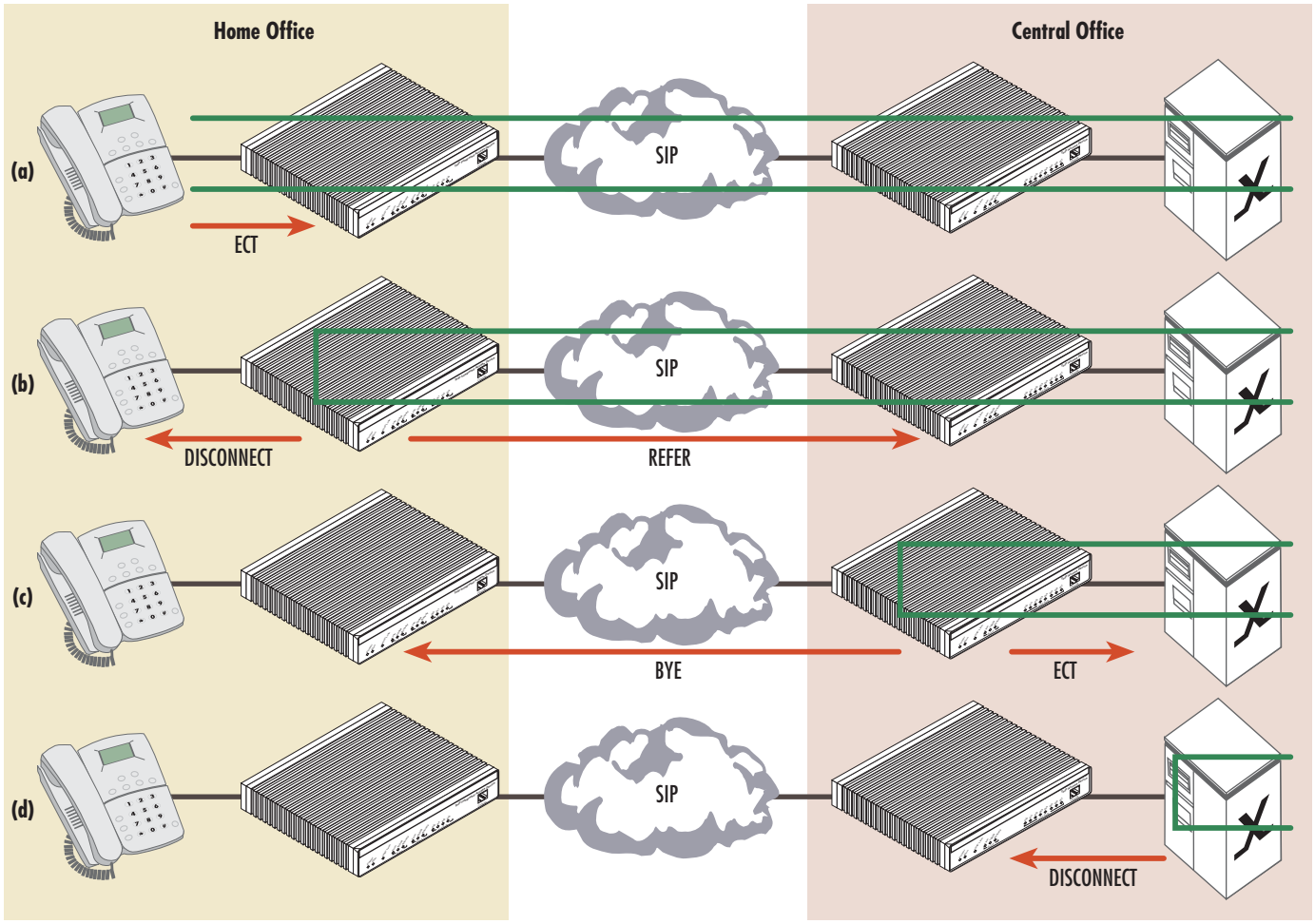


Figure 56. Example SIP network connecting two device to give a home office access to the CO PBX

The phone in the home office has two active calls to other subscribers of the PBX in the central office. The user wants to connect the other two participants and (**a**) sends an explicit call-transfer invocation to the device HO. The device HO internally connects the two calls and sends a DISCONNECT message to the phone for both calls. In a second step (**b**) the firmware on HO detects an internal loop. Both call legs are connected to the same network. In this example, both call legs are handled by the same SIP gateway. The firmware on device HO sends a REFER message to device CO, which connects the two call legs internally and sends a BYE message to the device HO. (**c**) Again the firmware of CO detect an internal loop. This time the call legs are handled by the same SIP interface, connected to the PBX. Since the ISDN port is a user port it sends an explicit call-transfer invocation to the PBX (**d**), which connects the call and sends the device CO a DISCONNECT message for both calls. During all these push back operations the datapath of the two participants keeps connected.

The push back mechanism over ISDN (using ECT) and SIP (using REFER) works independently of the protocol that invoked the call-transfer. For example, the same scenario also works if the phone in the home office is connected to an FXS port.

The push-back mechanism can be configured on each interface separately. Per default push-back is enabled for ISDN and SIP interfaces. You only have to change the configuration if you don't want internally looped calls to be pushed back to the network. The configuration command [**no**] **call-transfer accept** configures if an incoming call-transfer request (e.g. ECT or REFER) shall be accepted. The configuration command [**no**] **call-transfer emit** configures if a call reaching the device over this interface and leaving the device over this interface shall be pushed back to the network, i.e. if a call-transfer request (ECT or REFER) shall be sent.

The following procedure disables the push-back mechanism on the ISDN interface connected to the PBX. No ECT invocation is sent when a call is detected that is looped internally.

Step	Command	Purpose
1	***node*(ctx-ip)[ctx-*name*]# interface isdn <if-name>**	Go to the ISDN interface, for which you want to disable the push back mechanism.
2	***node*(if-isdn)[if-*name*]# no call-transfer emit**	Disable the push back mechanism

The following procedure disables the push-back mechanism on a SIP interface. No REFER message is sent when a call is detected that is looped internally.

Step	Command	Purpose
1	***node*(ctx-ip)[ctx-*name*]# interface sip <if-name>**	Go to the SIP interface, for which you want to disable the push back mechanism.
2	***node*(if-sip)[if-*name*]# no call-transfer emit**	Disable the push back mechanism

ISDN Advice of Charge support

The exchange of "Advice of Charge" information is supported between two ISDN interfaces. The charge information can be transmitted and received over H.323. (See Chapter 38, "H.323 interface configuration" on page 431 for additional information on AOC-D support for H.323). Without configuration changes SmartWare tunnels the "Advice of Charge" information from an ISDN user interface to an ISDN network interface. However you can disable AOC-S, AOC-D or AOC-E separately on each interface.

The network sends tariff information about a call using AOC-S messages at call (S)etup time and during the call when the tariff changes. Then (D)uring the call, the network sends the current charge in AOC-D messages. Finally at the (E)nd of the call, the network sends the total charge in an AOC-E message encapsulated in the DISCONNECT or RELEASE message.

The following procedure disables the reception of AOC messages from the network on an ISDN user interface.

Step	Command	Purpose
1	***node*(ctx-ip)[ctx-*name*]# interface isdn <if-name>**	Go to the ISDN interface, for which you want to disable AOC
2	***node*(if-isdn)[if-*name*]# no aoc-s**	Disables the reception of AOC-S messages at call setup time
3	***node*(if-isdn)[if-*name*]# no aoc-d**	Disables the reception of AOC-D messages during the call
4	***node*(if-isdn)[if-*name*]# no aoc-e**	Disables the reception of AOC-E messages at the end of the call

AOC is a network option that can be disable or set to be active for all calls or only on a per-call basis. If your network provider offers AOC on a per-call basis the firmware needs to request AOC information on each outgoing call. The following procedure enables the reception of AOC messages on an ISDN user interface. Additionally the interface sends an AOC activation request for each outgoing call to the network.

Step	Command	Purpose
1	***node*(ctx-ip)[ctx-*name*]# interface isdn <if-name>**	Go to the ISDN interface, for which you want to enable AOC on a per-call basis
2	***node*(if-isdn)[if-*name*]# aoc-s explicit**	Enables the reception of AOC-S messages and sends an AOC-S activation request for each outgoing call
3	***node*(if-isdn)[if-*name*]# aoc-d explicit**	Enables the reception of AOC-D messages and sends an AOC-D activation request for each outgoing call
4	***node*(if-isdn)[if-*name*]# aoc-e explicit**	Enables the reception of AOC-E messages and sends an AOC-E activation request for each outgoing call

In default, an ISDN network interface provides AOC information to the connected phones only if available, i.e. only if the call is routed to an ISDN user interface that is connected to a network providing AOC information. The following procedure enables the transmission of AOC message on an ISDN network interface even if

there is no AOC information from the network. In that case a message containing the value *noChargeAvailable* is sent.

Step	Command	Purpose
1	***node*(ctx-ip)[ctx-*name*]# interface isdn <if-name>**	Go to the ISDN network interface, for which you want to enable AOC for all calls
2	***node*(if-isdn)[if-*name*]# aoc-s automatic**	Enables the transmission of AOC-S messages even if there is no tariff information from the network for all calls
3	***node*(if-isdn)[if-*name*]# aoc-d automatic**	Enables the transmission of AOC-D messages even if there is not charge information from the network for all calls
4	***node*(if-isdn)[if-*name*]# aoc-e automatic**	Enables the transmission of AOC-E message even if there is no charge information from the network for all calls

The following procedure enables the transmission of AOC message on a per-call basis. That is AOC messages are sent by the connected phone only if configured for a per-call basis.

Step	Command	Purpose
1	***node*(ctx-ip)[ctx-*name*]# interface isdn <if-name>**	Go to the ISDN network interface, for which you want to enable AOC on a per-call basis
2	***node*(if-isdn)[if-*name*]# aoc-s explicit**	Enables the transmission of AOC-S messages even if there is no tariff information from the network on a per-call basis
3	***node*(if-isdn)[if-*name*]# aoc-d explicit**	Enables the transmission of AOC-D messages even if there is not charge information from the network on a per-call basis
4	***node*(if-isdn)[if-*name*]# aoc-e explicit**	Enables the transmission of AOC-E message even if there is no charge information from the network on a per-call basis

The following table shows an overview of the AOC variants:

	no aoc-*x*	aoc-*x* transparent	aoc-*x* automatic	aoc-*x* explicit
Default option	no	yes	no	no
ISDN User Interface (connected to a PBX switch etc.)				
No message from the network	No information forwarded to the peer interface	No information forwarded to the peer interface	No information forwarded to the peer interface	Sends an aoc-*x* request to the network. If the network rejects the request, no information is forwarded to the peer interface
AOC message from the network	No information forwarded to the peer interface	Information forwarded to the peer interface	Information forwarded to the peer interface	Information forwarded to the peer interface
ISDN Network Interface (connected to phones)				
Phone does not request AOC on a per-call basis	No information sent	Information sent as received from the network, no information sent if the network does not provide information	Always send information, *noChargeAvailable* sent if the network does not provide information	No information sent
Phone requests AOC on a per-call basis	No information sent	Information sent as received from the network, no information sent if the network does not provide information	Always send information, *noChargeAvailable* sent if the network does not provide information	Always send information, *noChargeAvailable* sent if the network does not provide information

ISDN DivertingLegInformation2 Facility

SmartWare is now able to extract the redirecting information from the DiverstingLegInformation2 Facility and to provide them to the call control. In the other direction, the redirecting information can be sent as DiverstingLegInformation2 Facility in addition to the Redirecting Number Information Element.

Transmit Direction

Mode: interface isdn <interface>

Step	Command	Purpose
1	[*name*] (**if-isdn**)[*interface*]**#[no] diversion emit**	Enables or disables transmitting of the DivertingLegInformation2 Facility.

Receive Direction

Mode: interface isdn <interface>

Step	Command	Purpose
1	[*name*] (**if-isdn**)[*interface*]**#[no] diversion accept**	Enables or disables receiving of the DivertingLegInformation2 Facility.

T1 Caller-Name Support

The ISDN implementation now supports reception and transmission of the caller-name on T1 links as it is used in NI2 networks according to Bellcore GR-1367-CORE. Transmission of the caller-name is part of the Calling Name Delivery (CNAM) service.

In previous build series (R3.20), the caller-name was already supported for DSS-1 networks using User-User information elements and for Q.SIG (PSS-1) networks using FACILITY messages. Now the caller-name is also supported for NI2 networks following the Bellcore standard.

As a prerequisite, the *caller-name* feature must be enabled on each ISDN interface in the CS context separately. This command now has additional arguments to configure the SETUP retention as follows:

In NI2 networks an incoming ISDN SETUP message may contain a *NameInfomationFollowing* indication instead of the name. This means that the calling-party name is not available yet, but will be sent later, for example, after the dictionary database lookup in progress succeeded. If such an incoming ISDN call is internally routed to another network (e.g. to a SIP network or to a ISDN DSS-1 network), we must know the name before sending the initial INVITE or SETUP message towards the destination network. Therefore we must retain the SETUP message of the incoming ISDN call until the name is present. The caller-name command now allows you to configure the behaviour of this SETUP retention mechanism. There are three possible options:

- **caller-name ignore-absence <timeout>:** This configuration command specifies the behaviour for incoming ISDN calls. When a *NameInformationFollowing* indication is received with the SETUP message, the call-initiation is retained until the name is received or until this timeout elapses. After that, the call is forwarded to the configured destination interface. When forwarding a call without a caller-name to a SIP network, please note that there is no chance to send the caller-name later over SIP.

- **caller-name early-alerting <timeout>**: This configuration command specifies the behaviour for incoming ISDN calls. Some networks only deliver the name after an alerting indication. These networks simulate the mid-ring name delivery feature of analog lines. If early alerting is enabled, we send back a faked ALERTING message after a configurable timeout when we receive a *NameInformationFollowing* indication. This command can be used together with the ignore-absence command. For example, you can configure an interface to first generate an ALERTING message and later forward the call anyway. If used that way, the early-alerting timeout should be smaller than the ignore-absence timeout.
- **caller-name send-information-following**: This configuration command specifies the behaviour for outgoing ISDN calls. If there is no name from the originating network, the ISDN interface configured with this command sends a *NameInformationFollowing* indication to the remote side itself.

The following example enables and configures the caller-name feature on a T1 ISDN interface for incoming calls. If no name is present in the SETUP message, but the SETUP message contains the *NameInformationFollowing* indication, an ALERTING message is sent back after 500ms. If there is no name after additional 500ms the call is routed to the destination network anyway.

Mode: context cs / interface isdn

Step	Command	Purpose
1	***node*(if-isdn)#caller-name**	Enables reception of the caller-name.
2	***node*(if-isdn)#caller-name early-alerting 500**	**(optional)** If no name is present in an incoming ISDN call and if the incoming SETUP message contains the NameInformationFollowing indication, we send a fake ALERTING message after 500ms towards the caller. The SETUP message is retained for this period, i.e. the call is not forwarded to the configured destination. This step is optional. When not configured, an ALERTING message is faked after **2s by default**. You can disable faking an ALERTING message by using the "no" form of the command. **Note:** If the ignore-absence timeout is also configured, the early-alerting timeout should have a smaller value than the ignore-absence timeout.

Step	Command	Purpose
3	***node*(if-isdn)#caller-name ignore-absence 1000**	**(optional)** If no name is present in an incoming ISDN call and if the incoming SETUP message contains the NameInformationFollowing indication, we forward the call to the routing destination anyway after 1000ms (500ms after faking the ALERTING message in this example). This step is optional. When not configured, the call is forwarded after **4s by default.** You can disable forwarding a call without a name by using the "no" form of the command. **Note:** The specified timeout is measured starting at the reception of the SETUP message, not when the early-alerting timeout elapses.

The following example enables and configures the caller-name feature on a T1 ISDN interface for outgoing calls. It enables the transmission of the NameInformationFollowing indication (encapsulated into sent SETUP message) when no name is present from the originating network:

Mode: context cs / interface isdn

Step	Command	Purpose
1	***node*(if-isdn)#caller-name**	Enables transmission of the caller-name.
2	***node*(if-isdn)#caller-name send-information-following**	If no name has been received from the originating network a NameInformationFollowing indication is send encapsulated into the SETUP message for the outgoing ISDN call. **This feature is disabled by default.**

Chapter 35 FXS interface configuration

Chapter contents

Introduction 405
FXS interface configuration task list 405
Configuring a subscriber number (recommended) 405
Configuring an alternate PSTN profile (optional) 406
Configuring caller-ID presentation (optional) 406
Configuring flash hook processing (optional) 406
Configuring ringing-cadence (optional) 407
Configuring the Message Waiting Indication feature for FXS 408
Configuration 408
Frequency-shift keying 409
FXS supplementary services description 410
Call hold 411
Call waiting 411
Call waiting reminder ring 412
Drop passive call 412
Drop active call 412
Call toggle 412
Call transfer 412
Conferencing 413
Call park 413

Introduction

This chapter provides an overview of FXS interfaces, and the tasks involved in their configuration. This chapter does not explain the basic configuration steps equal to all CS interfaces. Information about basic interface configuration can be found in the general chapter about CS interface configuration (see chapter 33, "CS interface configuration" on page 381).

An FXS interface represents the connection of an analog FXS port signaling to the call control of SmartWare. It encapsulates the signaling of the exchange side of a FXS line, allows incoming and outgoing calls on this line, controls the line events, tones and datapath, and provides a set of supplementary services. There is a one-to-one relation between the port and the interface: Only one port can bind to an existing interface, and there must be a port that binds to the interface for the interface to become functional (see figure 57).

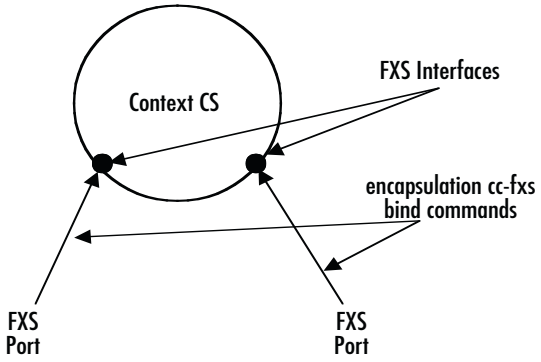


Figure 57. FXS interfaces on the CS context

FXS interface configuration task list

This section describes the configuration tasks for FXS interfaces. There is no mandatory configuration for basic FXS operation.

- Configuring a subscriber number (recommended) (see page 405)
- Using an alternate PSTN profile (optional) (see page 406)
- Configuring caller-id presentation (optional) (see page 406)
- Configuring flash hook processing (optional) (see page 406)

Configuring a subscriber number (recommended)

Contrary to ISDN, where each terminal knows its own subscriber number (MSN), an analog device doesn't have this capability. If such a device is connected to an FXS port and makes an outgoing call (goes off hook and dials), the dialed digits form the called party number. But there is no calling party information available from the FXS protocol. To insert calling party information and make it available to other protocols over which the call may be transported, a subscriber number must be configured on the interface.

Note The configured subscriber number does not affect the routing of incoming calls on the interface.

Mode: Interface FXS

Step	Command	Purpose
1	**[name](if-fxs)[name]#[no] subscriber-number** *<number>*	Applies a subscriber number for the fxs interface. The no form of the command clears an existing one. *Default:* none

Configuring an alternate PSTN profile (optional)

The PSTN profile contains the configuration for data/voice transmission on circuit-switched channels (See chapter 48, "PSTN profile configuration" on page 597). In the case of FXS interfaces, the PSTN profile applies to the analog line associated with the interface. There is a PSTN profile named default, which always exists in the system. If no different PSTN profile name is explicitly configured on the FXS interface, the profile named *default* is used.

Mode: Interface FXS

Step	Command	Purpose
1	**[name](if-fxs)[name]#use profile pstn** *profile-name*	Defines an alternate PSTN profile to be used for this FXS interface.

Configuring caller-ID presentation (optional)

FXS/FXO protocols allow the presentation of the caller-ID (calling party number and name of an incoming call) to an analog terminal when the terminal is ringing. (See Chapter 43, "FXS port configuration" on page 537 for other caller-ID related settings.) Use this command to configure presentation of the calling party number to the analog device connected to the FXS port associated with the interface.

Mode: Interface FXS

Step	Command	Purpose
1	**[name](if-fxs)[name]#[no] caller-id-presentation {pre-ring | mid-ring}**	Enables/Disables the caller-id presentation and defines if it must be sent before ring starts or in the first ring pause.

Configuring flash hook processing (optional)

The **flash-hook** command specifies if the flash hook pattern must be handled locally or if it must be relayed to the remote subscriber. Per default flash hook is handled locally because it is the initial pattern for local initiated supplementary services. If the flash hook is relayed to the remote subscriber, the local user will not be able to initiate a second call. The relay feature is used if the termination instance of the remote protocol is configured to handle all the supplementary services.

Mode: Interface FXS

Step	Command	Purpose
1	**[name](if-fxs)[name]#flash-hook {handle-locally | relay}**	Defines processing instance of the flash hook pattern. *Default:* handle-locally

Configuring ringing-cadence (optional)

The ringing-cadence on FXS Ringing-cadence profiles can be used on FXS interfaces. Each profile consists of a sequence of different rings and ring pauses. Arbitrary ring cadences of up to 10 elements can be configured. The sequence in which the commands are entered (or appear in the config file) defines the sequence in which the corresponding elements are played. If the profile consists of only one element, the element will be repeated forever, causing an endless ring or pause. An empty ringing-cadence will lead to an endless pause.

The profile ringing-cadence must be bound from the respective interface to become effective. Alternatively the profile ringing-cadence default can be modified to modify the ringing-cadence on all FXS interfaces.

Mode: Configure

Step	Command	Purpose
1	**[name](cfg)#profile ringing-cadence** *name*	Creates a ringing-cadence profile with name *name* and enters ringing-cadence configuration mode.
2	**[name](pf-ringingcad)[name]#play <duration>**	Defines a ring with duration *duration*.
3	**[name](pf-ringingcad)[name]#pause <duration>**	Defines a pause with duration *duration*.
4	**[name](pf-ringingcad)[name]#...**	Repeat step 2 and/or step 3 to define a ringing-cadence. If not specified otherwise, the new entries are appended to the existing cadence.
5	**[name](pf-ringingcad)[name]#flush-elements**	Resets the ringing cadence. Same as deleting and re-creating the profile.

Example: Define an example ringing-cadence

The first line defines the first element of the cadence: 500ms. The second line a pause of 500ms and so on. The cadence is repeated infinitely.

```
node(cfg)#profile ringing-cadence example
node(pf-ringingcad)[example]#play 500
node(pf-ringingcad)[example]#pause 500
node(pf-ringingcad)[example]#play 500
node(pf-ringingcad)[example]#pause 3000
```

Configuring the Message Waiting Indication feature for FXS

Note Message Waiting Indication is programmed in two sections of SmartWare, the FXS interface and the SIP Location service. The information below refers to information for configuring the Message Waiting Indication feature for FXS. For information on configuring the Message Waiting Indication feature for SIP, see "Configuring the Message Waiting Indication feature for SIP" on page 619.

FXS interface configuration mode allows a selection of the way in which notifications about new voice messages are performed. Supported modes are:

1. Stuttered dial tone (if there is a new message/messages in users voice mailbox, for 3 seconds after taking the phone off-hook the dial-tone will "stutter", producing short delays between the dial-tone cadence)
2. Visual Message Waiting Indication utilizing frequency shift keying signaling. Visual MWI-enabled FXS phones have a LED on the case to represent a new voice message. The led will blink (or will simply light up) when a new message arrives to the voice mailbox.
3. Both Stuttered dial tone from p.1 and Visual Message Waiting Indication from p.2.

Configuration

You can enable both stutter dial tone and visual message waiting indication at the same time issuing both **message-waiting-indication** commands independently.

Step	Command	Purpose
1	[*node*]**(if-fxs)[name]#[no] message-waiting-indication stutter-dial-tone**	Enables/Disables Message Waiting Indication through Stuttered Dial Tone

Step	Command	Purpose
1	[*node*]**(if-fxs)[name]#[no] message-waiting-indication frequency-shift-keying**	Enables Visual Message Waiting Indication through frequency shift keying signaling

Frequency-shift keying

Frequency-shift keying signaling can be specified in FXS port configuration mode, in one of the following formats:

1. Multiple Data Message Format (MDMF) (etsi standard) (default).
2. Single Data Message Format (SDMF) (bell standard).

To enable frequency-shift keying by MDMF standard, use the existing **caller-id format** command.

Note MDMF(etsi) is a default value and it will not be visible in "**show running-config**" output.

Step	Command	Purpose
1	**[*node*](prt-fxs)[num/num]#caller-id format etsi**	Selects MDMF standard for sending frequency-shift keyed messages

To enable frequency-shift keying by SDMF standard, use existing **caller-id format** command (please note that as of May 21st, 2008 there were **no confirmed test results** about SDMF frequency-shift keying signaling functioning properly):

Step	Command	Purpose
1	**[*node*](prt-fxs)[num/num]#caller-id format bell**	Selects SDMF standard for sending frequency-shift keyed messages

Note The tone cadence for stutter dial tone can be set in **profile tone-set** mode. The tone type name for a tone that should represent a waiting message is **message-waiting-tone** and by default it is set to **defaultStuttertone** (on,100,425,-7,off,100). Refer to Chapter 42, "Tone configuration" on page 529 for information on setting up tone profiles.

FXS supplementary services description

FXS interfaces offer a set of supplementary services. Some of these services are locally terminated. Others, like conferencing, push the involved subscribers to an external service. These supplementary services are:

- Call hold (see page 411)
- Call waiting (see page 411)
- Call waiting reminder ring (CWRR) (see page 411)
- Drop passive call (held or waiting) (see page 412)
- Drop active call (connected call), accept passive call (held or waiting) (see page 412)
- Call toggle (see page 412)
- Call transfer (see page 412)
- Conferencing (see page 413)
- Call park (see page 413)

Note The local subscriber is always the one who executes the service.

The services call hold and call waiting allow the local subscriber to open or accept a second call. This is the condition for all the other services that are applied as soon as the local subscriber has two calls initiated to remote subscribers. That means the call hold service can only be applied as long as only one call is ongoing. All the above described service can be executed by a key pattern. Some of these services have permanent patterns for the others they can be configured by the user.

Note The flash pattern always has the short notation, '!'.

Table 14. FXS services with permanent patterns

Service	Default Pattern
Call hold	!
Call transfer	On-Hook

Table 15. FXS services with configurable patterns

Service	Default Pattern
Drop passive call	!0
Drop active call, accept passive call	!1
Call toggle	!2
Conferencing	!3
Call park	*98

The command to change the key patterns is available in the interface fxs configuration mode. If the patterns start with the same key sequence, they should also have the same length. The key parser is of type best matching so it executes the service of the first pattern that represents a full match.

Mode: Interface FXS

Step	Command	Purpose
1	**[name](if-fxs)[name]#[no] {drop-passive | drop-active | toggle | coference}** *<pattern>*	Defines a new key pattern for the selected service. The no form of the command clears the existing pattern and disables the service. Use '!' for flash in a service pattern.

Call hold

This service is used to set the remote subscriber on hold and offers the possibility to set up a second call to the local subscriber. Call hold is activated by flash and then the local subscriber will hear the dial tone and the remote subscriber will be set on hold. For the action on the remote side, the far end device is responsible. It can play the hold tone, hold music or an inband message. In this state, the held subscriber can be retrieved by touching flash. If the local subscriber hears the dial tone and doesn't start dialing, the call waiting tone will play after a timeout of five seconds. To retrieve the held subscriber in this state, touch *flash*. It is possible to disable this service by executing the command no additional-call-offering on the fxs interface.

Mode: Interface FXS

Step	Command	Purpose
1	**[name](if-fxs)[name]#[no] additional-call-offering**	Enables or disables the call hold feature. *Default:* enabled

Call waiting

The call waiting service doesn't need a special pattern. The user can use either drop passive call to reject the incoming call, can use call toggle to hold the active call and to accept the incoming call or can use drop active call to disconnect the current one and to switch to the incoming call. Call waiting can be explicitly disabled by executing the no call-waiting command on the fxs interface.

The user of the device connected to the FXS port can be given the option to activate/deactivate call waiting by means of a special digit sequence touched on the keypad of his device (see section, "Configuring the precall service tables" on page 388). The configuration in the fxs interface is administrative what means if call waiting is disabled here, the user cannot activate it anymore.

Mode: Interface FXS

Step	Command	Purpose
1	**[name](if-fxs)[name]#[no] call-waiting**	Enables or disables the call-waiting service *Default:* enabled

Call waiting reminder ring

If a remote subscriber disconnects the call, the local subscriber will hear the release tone and go on-hook. But, if there is still a pending remote subscriber that has been previously set on hold, the phone rings right after on-hook. If the local subscriber goes off-hook, the user will be connected again with the previously held remote subscriber. It is possible to navigate to the pending held subscriber. As soon as the release tone appears, the local subscriber can touch *flash* to connect again.

Drop passive call

If the local subscriber hears the call waiting tone that indicates a second incoming call and decides not to disturb the current one, the drop passive call feature rejects the waiting call. It can also be used to drop a pending call that has been previously set on hold.

Drop active call

The drop active call service can be used if the local subscriber has an active call and a call previously set on hold. This feature provides the possibility to drop the active call and to switch to the hold one without going on-hook first.

Call toggle

In general, the call toggle feature is used to switch between an active and a passive call. The passive call can either be a call previously set on hold or a waiting call.

Call transfer

The call transfer feature is only available if the second call was originated by the local subscriber. For execution, the local subscriber goes on-hook and the active call and the held call will be transferred together. If the second call was an incoming call, the active call will be dropped and the CWRR will appear to signal a pending hold call. It is possible to disable this service by executing the command no call-transfer on the fxs interface.

Mode: Interface FXS

Step	Command	Purpose
1	**[name](if-fxs)[name]#[no] call-transfer**	Enables or disables the call transfer feature. *Default:* enabled

Conferencing

When executing a conference, all involved calls will be handed to a given call-router conference service. It is then the responsibility of this service to initiate the conference. Conference services are configured in the Context CS and are protocol specific. The sip-conference service can be used to address a SIP Media Server for conferencing according to RFC4240 (see section, "SIP conference-service" on page 525).

Conferencing can only be executed if the second call was originated by the local subscriber. If the second call is an incoming call and the local subscriber tries to conference, the call toggle service will be executed.

Mode: Interface FXS

Step	Command	Purpose
1	**[name](if-fxs)[name]#[no] route conference dest-service** *<service-name>*	Specifies the routing destination for a conference. The no form of the command clears an existing route.

Call park

The call park service allows parking an ongoing call on a specific number. This park number starts with a specific Park Code that must be configured with the 'service-pattern' command. After the user presses 'flash' to open a new call and the dialed number starts with the defined park service-pattern, the service will be executed five seconds after the last digit has been pressed or the user completes the number with the '#' character. This service is not handled locally, which means that another entity must be contacted that offers it.

On some SIP servers, it is possible to park a call on a specific number by using the blind-transfer feature. This is what the FXS park-call service achieves in an FXS/SIP call. It forces the SIP endpoint to execute a blind transfer where the Refer-To header contains the park number.

To change the park service pattern, the already existing 'service-pattern' command has the option 'park' that allows the user to overwrite the default setting.

Mode: Interface FXS

Step	Command	Purpose
1	**[name](if-fxs)[name]#[no]** {drop-passive | drop-active | toggle | conference | **park**} **<pattern>**	Defines a new key pattern for the selected service. The no form of the command clears the existing pattern and disables the service. Use '!' for flash in a service pattern.

Chapter 36 FXO interface configuration

Chapter contents

Introduction 415
FXO services description 416
Creating an FXO interface 416
Deleting an FXO interface 417
FXO interface configuration task list 418
- FXO off-hook on caller ID 418
- Configuring an alternate PSTN profile (optional) 418
- Configuring when the digits are dialed (optional) 419
- Configuring the number of rings to wait before answering the call (optional) 421
- Configuring how to detect a call has disconnected (optional) 422
- Configuring how to detect an outgoing call is connected (optional) 423
- Configuring the destination of the call 424
- FXO Mute dialing 424
- FXO interface examples 425

Introduction

This chapter provides an overview of FXO interfaces and the tasks involved in configuring them. This chapter does not explain the basic configuration steps common to all Context Switch (CS) interfaces. Information about basic interface configuration can be found in chapter 33, "CS interface configuration" on page 381.

An FXO, *Foreign eXchange Office*, interface connects to an FXS, *Foreign eXchange Subscriber*, interface. These two interfaces are used in analog telephony. The FXS interface is provided at the central office in order to connect to telephones, modems, PBXs, faxes, etc. Telephones and modems are FXO interfaces and want to connect to the central office. The FXO interface in the SmartNode products is like the telephone and modem interface.

In SmartWare, an FXO interface functions to connect the analog FXO port's call signaling to the call control process in SmartWare. Recall that an interface in SmartNode products is a logical device and a port is a physical device. So the FXO feature consists of the logical interface with all its processes together with its configurable parameters and the physical interface for the actual analog, 2-wire connection to an FXS device. There is a one-to-one correspondence between the port and the interface.

In order for the interface to be able to make a connection over the 2-wire analog line, there must be a port bound to the interface (see figure 58). For more information on ports, interfaces, and binding, see section "Interfaces, Ports, and Bindings" on page 47. For specific details on binding the FXO port to an FXO interface, see section "Bind FXO ports to higher layer applications" on page 543.

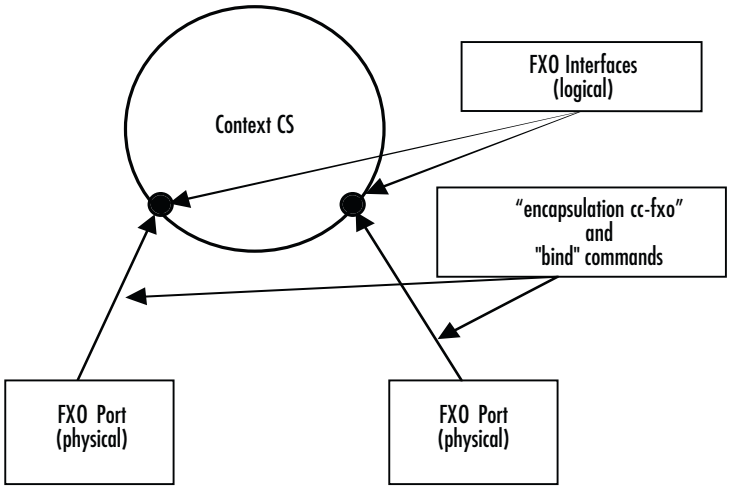


Figure 58. FXO interfaces on the CS context

This chapter includes the following sections:

- FXO services description (see page 416)
- Creating an FXO interface (see page 416)
- Deleting an FXO interface (see page 417)
- FXO interface configuration task list (see page 418)

FXO services description

The wide variety of applications and services are supported through a rich feature set. The major characteristics and features are

- 2-wire loop-start
- Off-hook and ring detection supervision
- Automatic and programmable line gain
- Programmable ring count before call pick-up
- End-of-call detection by line drop (call release indication), busy tone detection and battery reversal detection
- Hook-flash sending: programmable duration, H.245 hook-flash relay ("!" in user input) which provides Cisco compatibility
- DTMF send and detect: programmable interdigit timer, DTMF-relay
- Caller ID (CLID) and Caller ID names FSK command line interface reception and relay to VoIP signaling (Bellcore/ANSI and ETSI/ITU)
- Call routing based on caller ID
- Second dial-tone for two-stage DTMF dialing with call routing based on DTMF numbers.

Creating an FXO interface

Interface names can be any arbitrary string. Use self-explanatory names for your interfaces to reflect their usage in your application. After creating the FXO interface, it is necessary to bind the FXO interface. Refer to chapter 44, "FXO port configuration" on page 542 for details.

Mode: Context CS

You enter Context CS, one of the configuration modes, as follows.

Note *Node* is the host name you have assigned to your SmartNode and is the basic prompt.

Step	Prompt & command	Purpose
1	**node>**	Basic prompt in Operator Exec mode
2	**node>enable**	Enters Administration execution mode
	node#	The prompt in administration execution mode
3	**node#configure**	Enters the Configure configuration mode
	node(cfg)#	The prompt in the Configure configuration mode
4	**node(cfg)#context cs**	Enters the context "CS" for Circuit Switch
	node(ctx-cs)[switch]#	The prompt in the Context CS configuration mode

Once you are in the Context CS mode, you can enter the FXO configuration mode with the next steps.

Step	Prompt & command	Purpose
5	**node(ctx-cs)[switch]#interface fxo** *name*	The "interface fxo" command creates the new interface name, which represents an FXO interface. This command also places you in the FXO interface configuration mode for the created interface.
6	**node(if-fxo)**[*name*]**#**	You are now in the FXO interface configuration mode. In this mode, you may configure the parameters for the FXO interface name

Example: Create an FXO interface named *PSTN-FALLBACK*

The following commands would be used—in Context CS mode—to create an FXO interface named *PSTN-FALLBACK*:

```
node(ctx-cs)[switch]#interface fxo PSTN-FALLBACK
node(if-fxo)[PSTN-FA~]#
```

Deleting an FXO interface

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

Mode: Context CS.

Step	Prompt & command	Purpose
1	**node>**	Basic prompt in Operator Exec mode
2	**node>enable**	Enters Administration execution mode
	node#	The prompt in administration execution mode
3	**node#configure**	Enters the Configure configuration mode
	node(cfg)#	The prompt in the Configure configuration mode
4	**node(cfg)#context cs**	Enters the context "CS" for Circuit Switch
	node(ctx-cs)[switch]#	The prompt in the Context CS configuration mode
5	**node(ctx-cs)[switch]#no interface fxo** *name*	Deletes the existing interface *name*

FXO interface configuration task list

There are numerous configurable parameters that apply to the FXO interface. The basic commands are listed with a short description of their function.

- **ring-number on-caller-id**—Determines if the FXO interface will go off-hook upon reception of a specified caller-ID.
- **use profile pstn**—Defining and applying an alternate PSTN profile for a specific FXO interface
- **dial-after**—Selecting whether the FXO interface dials after a pre-defined time or after detection of dial-tone
- **ring-number**—Defining how many rings are received before answering an incoming call
- **disconnect-signal**—Selecting the method of determining a call has been disconnected
- **connect-signal**—Choosing how to detect the connection on the remote end of an outgoing call
- **route**—Determining the destination (interface) for the incoming call

FXO off-hook on caller ID

A new option has been added to the command "ring-number". Instead of specifying the number of ring bursts to wait before going off-hook (for calls coming in through FXO), it is now also possible to go off-hook upon reception of the caller-ID. With this setting, if a caller ID is available, the time before FXO goes off-hook to accept the call can be decreased by 2 to 3 seconds. If no caller ID is detected, the call is accepted upon reception of the second ring-burst.

Mode: Interface FXO

Step	Prompt, command & response	Purpose
1	[*name*] **(if-fxo)# ring-number on-caller-id**	Accepts a call coming in through FXO after reception of the caller ID or the second ring burst. Default: 1 ring.

Configuring an alternate PSTN profile (optional)

The PSTN profile has the following configurable parameters:

- Echo canceller (can be enabled or disabled)
- Output gain, which sets the volume of the output of the PSTN interface and port, in this case, FXO.

To define an alternate PSTN profile for the FXO interface, first create the profile according to instructions in chapter 48, "PSTN profile configuration" on page 597. Then you can apply the newly defined PSTN profile to a specific FXO interface with the **use** command as follows.

First enter the Interface FXO configuration mode.

Mode: Interface FXO

Step	Prompt, command & response	Purpose
1	**node>**	Basic prompt in Operator Exec mode
2	**node>enable**	Enters Administration execution mode
	node#	Response: The prompt in administration execution mode is the #
3	**node#configure**	Enters the Configure configuration mode
	node(cfg)#	Response: The prompt in the Configure configuration mode is (cfg)#
4	**node(cfg)#context cs**	Enters the Context CS configure mode
	node(ctx-cs)[switch]#	Response: The prompt in the Context CS configuration mode is (ctx-cs)[switch]#
5	**node(ctx-cs)[switch]#interface fxo** *if-name*	Enter the Interface FXO configuration mode
	node(if-fxo)*[if-name]***#**	Response: The prompt in the Interface FXO configuration mode is (if-fxo)*[if-name]*#

Now we can apply the PSTN profile for the FXO interface named *name* as follows.

Step	Prompt, command & response	Purpose
6	**node(if-fxo)[if-name]#[no] use profile** pstn profile-name	The "profile pstn" command is applied to the FXO interface named *if-name*
	node(if-fxo)*[if-name]***#**	Response: You are now in the FXO interface configuration mode. In this mode, you may configure the parameters for the FXO interface *name*.

Configuring when the digits are dialed (optional)

When the FXO port goes off-hook to make an outgoing call, the FXS switch normally sends a dial-tone to indicate it is ready to received dialed digits. Alternatively, you can specify the FXO interface to wait a specific period of time after going off-hook before dialing the first digit.

> **Note** All countries do not have the same dial-tone. For information on configuring the dial-tone for your country, refer to chapter 42, "Tone configuration" on page 529.

The default setting is to wait for the dial-tone. If you choose to wait a specific time after going off-hook, the variable is timeout and specify the number of seconds to wait. Zero (0) seconds means that the interface dials immediately.

Mode: Interface FXO

Step	Prompt, command & response	Purpose
1	**node>**	Basic prompt in Operator Exec mode
2	**node>enable**	Enters Administration execution mode
	node#	Response: The prompt in administration execution mode is the #
3	**node#configure**	Enters the Configure configuration mode
	node(cfg)#	Response: The prompt in the Configure configuration mode is (cfg)#
4	**node(cfg)#context cs**	Enters the Context CS configure mode
	node(ctx-cs)[switch]#	Response: The prompt in the Context CS configuration mode is (ctx-cs)[switch]#
5	**node(ctx-cs)[switch]#interface fxo** *if-name*	Enter the Interface FXO configuration mode
	node(if-fxo)*[if-name]***#**	Response: The prompt in the Interface FXO configuration mode is (if-fxo)*[if-name]*#
6	**node(if-fxo)***[if-name]***#dial-after {dial-tone | timeout** *seconds***}**	Specifies whether to dial after detection of dial-tone (default) or to wait for a specified timeout (in seconds). Zero (0) seconds will initiate dialing immediately after going off-hook.

Example: Setting the timeout to be 4 seconds after going off-hook when initiating a call. The timeout is set for the FXO interface named *Line0*.

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface fxo Line0
node(if-fxo)[Line0]#dial-after timeout 4
```

You can verify the change in configuration by using the **show running-config** command.

This is one of the few commands that does not have a **no** inverse operation of the command. If you want to change the timeout period, re-enter the command with the new timeout period. The other option is to wait for dial-tone. To return to the default (waiting for dial-tone before dialing), enter the command again using **dial-tone**.

Note Verify that you have configured the dial-tone for the country in which the SmartNode is installed. (see chapter 42, "Tone configuration" on page 529). If the dial-tone is not configured for the proper country, the FXO interface will not detect when the remote FXS switch is sending dial-tone.

Configuring the number of rings to wait before answering the call (optional)

An FXO port identifies an incoming call by detecting the ring from the FXS switch. The **ring-number** is a configurable parameter which selects the number of rings before answering the incoming call, that is, before going off-hook and establishing the call. The minimum value for **ring-number** is zero (0). With a ring-number of zero, the FXO interface never answers an incoming call.

Due to variations between countries, the proper setting may be 1 or 2. In the USA the Caller-ID (CLID) is sent to the FXO port between the first and second ring, so a ring-number of 2 would be appropriate. On the other hand, numerous countries send the CLID prior to the first ring, so the default setting of 1 would be satisfactory.

Mode: Interface FXO

Step	Prompt, command & response	Purpose
1	**node>**	Basic prompt in Operator Exec mode
2	**node>enable**	Enters Administration execution mode
	node#	Response: The prompt in administration execution mode is the #
3	**node#configure**	Enters the Configure configuration mode
	node(cfg)#	Response: The prompt in the Configure configuration mode is (cfg)#
4	**node(cfg)#context cs**	Enters the Context CS configure mode
	node(ctx-cs)[switch]#	Response: The prompt in the Context CS configuration mode is (ctx-cs)[switch]#
5	**node(ctx-cs)[switch]#interface fxo** *if-name*	Enter the Interface FXO configuration mode
	node(if-fxo)*[if-name]***#**	Response: The prompt in the Interface FXO configuration mode is (if-fxo)*[if-name]*#
6	**node(if-fxo)***[if-name]***#ring-number** *count*	Specifies the number of rings to wait before going off-hook. Default = 1 ring.

Example: Configure the ring number to wait for CLID by setting the *count* to 2. The name of the specific FXO interface is *pstn-local*.

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface fxo pstn-local
node(if-fxo)[pstn-local]#ring-number 2
```

You can verify the change in configuration by using the **show running-config** command.

Configuring how to detect a call has disconnected (optional)

When a call has disconnected, the FXO interface may detect and verify the termination of the phone call by three different methods.

- Detect a busy tone or release tone
- Detect a loop break (if provided by the FXS switch)
- Detect battery reversal (if provided by the FXS switch)
- Detect a dtmf signal
- Configure loop break timing

The selection of the method, if any of the three, is via the **disconnect-signal** command. The default enables only loop-break. Upon detecting a *loop break*, the FXO interface proceeds to clear the call on the SmartNode. In some instances, the user may need to transmit all of the in-band information (tone signal, announcement) to the end party after disconnection has occurred.

The **disconnect-signal** command can be used to enable or disable the three detection methods. If all three methods are disabled, the call is cleared after a period of 30 seconds from reception of the disconnect signal. Consequently it becomes the responsibility of the end party to execute the on-hook (completing the disconnection phase) for the call to be completely cleared.

Note Verify that the busy and release tones are correctly configured for the country where the SmartNode is installed (see chapter 42, "Tone configuration" on page 529 for configuration information. If the tones are improperly configured, the FXO port will not detect them, resulting in missed phone calls.

Mode: Interface FXO

Step	Prompt, command & response	Purpose
1	**node(if-fxo)***[if-name]***#[no] disconnect-signal {battery-reversal | busy-tone | loop-break}**	The default is Loop-break. To disable it, use the no inverse command. Should all three methods be disabled, the call is cleared 30 seconds after receiving the disconnect signal. The default setting of loop-break is not displayed in the running-config output.
2	**[***name***] (if-fxo) [***interface***]# [no] disconnect-signal dtmf <dtmf-digits>**	Disconnects the call upon reception of one of the specified DTMF digits.
3	**[***name***] (if-fxo) [***interface***]#loop-break-duration [min <min-time>] [max <max-time>]**	Specifies the timing of the loop break signal. Shorter breaks are ignored, longer breaks are treated as line down. Min specifies the minimum time of a valid loop break (Default 60ms). Max specifies the maximum time of a valid loop break (Default 600ms). Please note that the internal timer has a resolution of 40ms.

Note Use the battery-reversal disconnect signal with caution, and use only if the battery-reversal connect signal is also enabled.

Configuring how to detect an outgoing call is connected (optional)

An FXO interface has the following methods for verifying the connection of an outgoing call after the dialing has been completed:

- Detect battery reversal (if provided by the FXS switch)
- Detect the first tax pulse (if provided by the FXS switch)

Note **Tax Impulse Signals:** European telephone companies in Austria, Belgium, Czechoslovakia, Germany, Spain and Switzerland place a *pulse signal* on the phone line to meter the length of the telephone call for billing purposes.

The command to enable or disable these methods is **connect-signal**. If both are enabled, only one needs to occur for the FXO interface to verify a properly connected call with the remote party. Should both be disabled, the SmartNode waits for the call-connect signal from the FXS switch.

Mode: Interface FXO

Step	Prompt, command & response	Purpose
1	**node>**	Basic prompt in Operator Exec mode
2	**node>enable**	Enters Administration execution mode
	node#	Response: The prompt in administration execution mode is the #
3	**node#configure**	Enters the Configure configuration mode
	node(cfg)#	Response: The prompt in the Configure configuration mode is (cfg)#
4	**node(cfg)#context cs**	Enters the Context CS configure mode
	node(ctx-cs)[switch]#	Response: The prompt in the Context CS configuration mode is (ctx-cs)[switch]#
5	**node(ctx-cs)[switch]#interface fxo** *if-name*	Enter the Interface FXO configuration mode
	node(if-fxo)*[if-name]***#**	Response: The prompt in the Interface FXO configuration mode is (if-fxo)*[if-name]*#
6	**node(if-fxo)***[if-name]***#[no] connect-signal {battery-reversal | tax-pulse}**	Selects battery-reversal, tax-pulse or neither to determine when outgoing calls are connected. Default: both methods are disabled.

Note Only disable connect-signal if you are sure that the FXS switch provides a call connect signal.

Configuring the destination of the call

The last command in configuring the FXO Interface is the route command. This command configures the call router. You can configure the routing-destination for call setup and for service activation. For complete details, see chapter 40, "Call router configuration" on page 456.

Mode: Interface FXO

Step	Prompt, command & response	Purpose
1	**node>**	Basic prompt in Operator Exec mode
2	**node>enable**	Enters Administration execution mode
	node#	Response: The prompt in administration execution mode is the #
3	**node#configure**	Enters the Configure configuration mode
	node(cfg)#	Response: The prompt in the Configure configuration mode is (cfg)#
4	**node(cfg)#context cs**	Enters the Context CS configure mode
	node(ctx-cs)[switch]#	Response: The prompt in the Context CS configuration mode is (ctx-cs)[switch]#
5	**node(ctx-cs)[switch]#interface fxo** *if-name*	Enter the Interface FXO configuration mode
	node(if-fxo)*[if-name]***#**	Response: The prompt in the Interface FXO configuration mode is (if-fxo) *[if-name]*#
6	**node(if-fxo)***[if-name]***#[no] route {call {dest-interface** *interface-name* **| dest-service** *table-name* **| dest-table** *service-name*} **| precall {dest-interface** *interface-name* **| dest-service** *table-name* **| dest-table** *service-name* **}**	Use this command to route a call (dest-interface) directly to an interface specified with the *interface-name* parameter, (dest-table) to the call router using the *table-name* table as the first routing table, or (dest-table) directly to a service specified with the *service-name* parameter.

FXO Mute dialing

A new command has been added. With this command, the FXO interface can mute its receive path during dialtone detection and DTMF digits sending. This to avoid unwanted noises on the calling side (for calls going out of the device through FXO).

Mode: interface fxo

Step	Command	Purpose
1	**[***name***] (if-fxo)# [no] mute-dialing**	Enables or disables mute of receive path during dialtone detection and dialing. Default: *Disabled*.

FXO interface examples

Example 1: Configuring an FXO interface which is to be connected to a PSTN network for analog line extension over IP. The FXS switch provides caller-id between the first and second ring and uses battery reversal to indicate a connected call. The FXO interface is named *pstn-local*. The incoming call is routed directly to the interface named *pstn-1-voip*.

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface fxo pstn-local
node(if-fxo)[pstn-local]#connect-signal battery-reversal
node(if-fxo)[pstn-local]#ring-number 2
node(if-fxo)[pstn-local]#route call dest-interface pstn-1-voip
```

Example 2: Configuring an FXO interface to be used as fallback if the IP network link is down. This means that there are only out-going calls. You are not sure whether the FXS switch provides a connect signal. In this case, you only have to create the interface and bind the FXO port to the FXO interface. (For binding the FXO port to the FXO interface, see chapter 44, "FXO port configuration" on page 542.)

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface fxo pstn-fb
node(if-fxo)[pstn-fb]#connect-signal battery-reversal
node(if-fxo)[pstn-fb]#connect-signal tax-pulse
```

Chapter 37 RBS interface configuration

Chapter contents

Introduction 427
RBS interface configuration task list 427
Creating/Deleting a RBS interface 427
- Configuring an alternate PSTN profile 427
- Configuring an alternate Tone-Set profile 428
- Configuring B-Channel allocation strategy 428
- Configuring additional disconnect signals 428
- Configuring number of Rings before Off-Hook 429
- Configuring ready to dial strategy 429
- RBS interface debugging 429

Introduction

This chapter provides an overview of RBS interfaces, and the tasks involved in their configuration. This chapter does not explain the basic configuration steps equal to all CS interfaces. Information about basic interface configuration can be found in the general chapter about CS interface configuration (see Chapter 33, "CS interface configuration" on page 381). An RBS interface represents the connection of a T1 timeslot or of a group of timeslots. For every timeslot bound to the interface exist a RBS protocol endpoint that masters incoming and outgoing calls, controls the B-channel and provides different services. A RBS interface can encapsulate subscriber and exchange side of the following protocols: Loop Start, Ground Start, E&M Immediate Start, E&M Wink Start, E&M Double Wink Start. The settings are automatically taken from the RBS protocol that binds to the interface and changes of the protocol configuration are automatically reflected on the interface. See Chapter 20, "RBS configuration" on page 225 for more details.

RBS interface configuration task list

- Creating/Deleting a RBS interface
- Configuring an alternate PSTN profile
- Configuring an alternate Tone-Set profile
- Configuring B-Channel allocation strategy
- Configuring additional disconnect signals
- Configuring number of Rings before Off-Hook
- Configuring ready to dial strategy

Creating/Deleting a RBS interface

Interface names can be any arbitrary string. Use self-explanatory names for your interfaces to reflect their usage in your application. After creating the RBS interface, it is necessary to bind the requested RBS protocol to it. See Chapter 20, "RBS configuration" on page 225 for more details.

Mode: Context CS

Step	Prompt & command	Purpose
1	**node(ctx-cs)[switch]#[no] interface rbs** *name*	The "interface rbs" command creates a new interface, the 'no' form deletes an existing one.

Configuring an alternate PSTN profile

The PSTN profile contains the configuration for data/voice transmission on circuit-switched channels (see Chapter 48, "PSTN profile configuration" on page 597). In the case of RBS interfaces, the PSTN profile applies to the B-Channels of the timeslots associated with the interface.

There is a PSTN profile named *default*, which always exists in the system. If no different PSTN profile name is explicitly configured on the RBS interface, the profile *default* is used.

Mode: Interface RBS

Step	Command	Purpose
1	***node*(if-rbs)[***if-name*]#use profile pstn** *profile-name*	Defines an alternate PSTN profile to be used for this RBS interface/Reverts the setting to its default (use profile PSTN *default*)

Configuring an alternate Tone-Set profile

The Tone-Set profile contains the mapping of the different Call Progress Tones like Dial-Tone, Ringback-Tone or Release-Tone to programmed tone sequences. Dependent on the configuration, the RBS protocols must be able to detect inband played Dial or Release Tones. Therefore it is important, both communication parties using the same tone specifications. For details how to setup and alternate Tone-Set profile, please consult chapter 42, "Tone configuration" on page 529.

There is a Tone-Set profile named *default*, which always exists in the system. If no different Tone-Set profile name is explicitly configured on the RBS interface, the profile *default* is used.

Mode: Interface RBS

Step	Command	Purpose
1	***node*(if-rbs)[***if-name*]#use profile tone-set** *profile-name*	Defines an alternate Tone-Set profile to be used for this RBS interface/Reverts the setting to its default (use profile Tone=-Set *default*)

Configuring B-Channel allocation strategy

If a group of timeslots is bound to the interface and a call is originated by the Call Control, the B-Channel allocation strategy defines if the highest available timeslot number must be chosen for initiating the outgoing call or the lowest one. For incoming calls over TDM this command has no effect because the timeslot has already been selected by the remote party.

Mode: Interface RBS

Step	Command	Purpose
1	***node*(if-rbs)[***if-name*]#bchan-number-order {ascending | descending}**	Defines the B-Channel allocation strategy Default: descending

Configuring additional disconnect signals

Most of the RBS protocols define a ABCD-Bit pattern can be sent to indicate a call disconnection to the remote party. In case of the Loop Start protocol where the exchange side terminates the call, this is not possible. This protocol is really similar to an analog telephony line where the subscriber will be informed with a release tone about a call release. With this command it is possible to configure additional signals must be interpreted as a disconnect event.

Mode: Interface RBS

Step	Command	Purpose
1	***node*(if-rbs)[*if-name*]#[no] disconnect-signal {busy-tone}**	Enables/Disables the busy/release-tone as additional disconnect signal. Default: Enabled

Configuring number of Rings before Off-Hook

The Loop Start and the Ground Start protocol on the subscriber side identifying an incoming call by detecting the Ring-Signal sent by the exchange side. This command specifies the number of ring cycles before the subscriber side is going Off-Hook and answers the call.

Mode: Interface RBS

Step	Command	Purpose
1	***node*(if-rbs)[*if-name*]#ring-number** *value*	Defines the number of ring cycles before the subscriber side answers the call. Default: 1

Configuring ready to dial strategy

If on the Loop Start or Ground Start protocol the subscriber side originates a call, there is no protocol specification for the exchanges side to signal readiness for accepting the called party number. Even the E&M Immediate Start protocol is symmetric, the terminating side is also unable to do that. This command specifies for these protocols the strategy they must apply to determine the right moment for sending the called party number.

- Dialtone: The originating side sends the called party number as soon as it detects the dial tone.
- Timeout: The originating side sends the called party number after a timeout that starts at the Off-Hook moment.

Mode: Interface RBS

Step	Command	Purpose
1	***node*(if-rbs)[*if-name*]#[no] dial-after {dial-tone | timeout** *seconds*}	Defines the ready to dial strategy. Default: dialtone

RBS interface debugging

For the investigation of possible call signaling or interoperability problems, there exists a debug command with the options ‘datapath’, ‘error’ and ‘signaling’. In addition exists a ‘show’ command that outputs information about the current interface configuration and about the states of the protocol endpoints.

Mode: Operator execution

Step	Command	Purpose
1	***node*#[no] debug ccrbs {datapath | error | signaling}**	Enables/Disables different RBS interface monitors.

Mode: Operator execution

Step	Command	Purpose
1	***node*#show ccrbs call** *if-name* **[detail** *level***]**	Prints information about ongoing calls on the selected interface.
	node*#show ccrbs interface** *if-name* **[detail** *level]**	Prints information about the configuration of the selected interface and about the states of the belonging protocol endpoints.

Chapter 38 H.323 interface configuration

Chapter contents

Introduction 432
H.323 interface configuration task list 432
Binding the interface to an H.323 gateway 433
Configuring an alternate VoIP profile (optional) 434
Configuring CLIP/CLIR support (optional) 435
Enabling 'early-proceeding' on H.323 interfaces 436
Enabling the early call connect (optional) 436
Enabling the early call disconnect (optional) 437
Enabling the via address support (optional) 437
Override the default destination call signaling port (Optional) 437
Configuring status inquiry settings (optional) 438
AOC-D Support for H.323 439

Introduction

This chapter provides an overview of H.323 interfaces used by H.323 gateways and describes the specific tasks involved in their configuration. This chapter does not explain the basic configuration steps required for all CS interfaces. Information about basic interface configuration can be found in the general chapter about CS interface configuration.

Within the CS context of SmartWare, an H.323 interface is a special type of CS interface providing call routing for incoming and outgoing calls to and from the H.323 gateway (see figure 59).

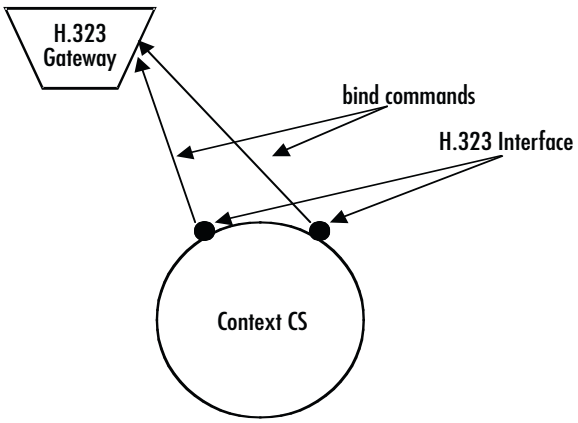


Figure 59. H.323 interfaces on the CS context

An H.323 interface is a CS interface type that also provides voice over IP settings in addition to the general CS interface parameters. All H.323 interfaces must be explicitly bound to an H.323 gateway. Calls, which are routed from the Context CS to one of the H.323 interfaces, will be forwarded for call establishment to the H.323 gateway to which the H.323 interface is bound. All the parameters configured in the H.323 interface will be applied to the forwarded call.

When a call arrives over H.323 in the H.323 gateway. The gateway looks for the best matching H.323 interface, which is bound to that gateway. If there is an H.323 interface, which contains the IP address of the source of the H.323 call in its *remoteip* configuration parameter, the call will be handed over to that interface for further call processing. If no such interface is found, the gateway looks for an interface, which is bound to that gateway and does not contain a *remoteip* parameter. If such an interface is found the call will be handed over to that interface for further processing. If however such an interface is also not available, the call will be dropped.

H.323 interface configuration task list

This section describes the configuration tasks for H.323 interfaces listed below. You must at least perform the tasks, which are not marked as optional, to define a working H.323 interface. The optional tasks are usually only required in advanced configurations. Before you can start with the H.323 interface specific configuration tasks, you need to create the H323 interface and define the routing for it as defined in chapter 33, "CS interface configuration" on page 381.

- Binding the interface to an H.323 gateway
- Configuring a remote IP address
- Using an alternate VoIP profile (optional)

- Configuring information transfer capability handling (optional)
- Configuring CLIP/CLIR support (optional)
- Enabling the early-proceeding feature for call setup
- Enabling the early call disconnect (optional)
- Enabling the via address support (optional)
- Overriding the default destination call-signaling port (optional)
- Configuring status inquiry settings (optional)

Binding the interface to an H.323 gateway

Every H.323 interface must be explicitly bound to an H.323 gateway instance.

Procedure: To bind an H.323 interface to an H.323 gateway.

Mode: Interface H.323

Step	Command	Purpose
1	***node*(if-h323)[*if-name*]#bind gateway** *gw-name*	Binds the gateway to an H.323 gateway.

Examples: Bind the H.323 interfaces to an H.323 gateway instance

The following example shows how to bind an H.323 interface named *MyH323If* to an H.323 gateway instance named *h323*.

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface h323 MyH323If
node(if-h323)[MyH323If]#bind gateway h323
Configure a remote IP address
```

If the gateway to which the H.323 interface is bound does not use a gatekeeper, it is required to specify the IP address of the remote entity for which the H.323 interface is used directly within the H.323 interface. This is done using the procedure below. If the H.323 gateway however uses a gatekeeper, the gatekeeper is responsible for resolving the remote entities IP address. In that case this procedure must not be used.

Procedure: To specify the remote H.323 entities IP address in the H.323 interface.

Mode: Interface H.323

Step	Command	Purpose
1	***node*(if-h323)[*if-name*]#remoteip** *ip-address*	Defines the IP address of the remote H.323 entity, for which this interface shall be used.

Examples: Define the IP address of the remote H.323 entity

The following example shows how to associate an H.323 interface named *MyH323If* with a remote H.323 entity, which has the IP address 1.2.3.4

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface h323 MyH323If
node(if-h323)[myh323if]#remoteip 1.2.3.4
```

Configuring an alternate VoIP profile (optional)

Normally, the VoIP profile defined in the H.323 gateway is used for all the calls over that gateway. However, it is possible to specify an alternate VoIP profile in the H.323 interface. In that case the VoIP profile defined within the VoIP interface is used for all the calls established using that H.323 interface instead of the VoIP profile defined in the H.323 gateway.

Procedure: To define an alternate VoIP profile for the H.323 interface

Mode: Interface H.323

Step	Command	Purpose
1	***node*(if-h323)[*if-name*]#use profile voip** *profile-name*	Defines an alternate VoIP profile to be used for this VoIP interface

Example: Configure an alternate VoIP profile

The following example shows how to replace the default VoIP profile of the H.323 gateway with a VoIP profile named *myprofile* for an H.323 interface named *MyH323If*.

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface h323 MyH323If
node(if-h323)[MyH323If]#use profile voip myprofile
Configure information transfer capability handling (Optional)
```

Normally, the H.323 gateway transparently forwards the information transfer capability information element between the H.323 network and other gateways of the SmartNode. There are, however, several H.323 clients that do not provide correct information transfer capability information. One of the most often used clients of this type is Microsoft Netmeeting. When communicating to one of these clients, it is necessary to define the correct information transfer capability in the H.323 interface. It is possible to define for each direction (for calls from or to H.323) separately, whether the information transfer capability received from the network, or another information transfer capability should be used for the calls.

Note The default behavior when not configured otherwise is to set the information transfer capability of incoming calls to 3k1-audio and to transparently pass the information transfer capability for outgoing calls.

Procedure: To configure information transfer capability overriding

Mode: Interface H.323

Step	Command	Purpose
1	***node*(*if- h323*)[*if-name*]#itc rx {3k1-audio | 7k-audio | restricted-digital | unrestricted-digital | speech | video | transparent }**	Specifies the information transfer capability to be used for calls from the H.323 gateway to another gateway of the system (incoming calls). All settings force the specified information transfer capability to be used except for the transparent setting, which indicates that the information transfer capability of the call should be forwarded transparently.
2	***node*(*if- h323*)[*if-name*]#itc tx {3k1-audio | 7k-audio | restricted-digital | unrestricted-digital | speech | video | transparent }**	Specifies the information transfer capability to be used for calls from any gateway of the system to the H.323 gateway (outgoing calls). All settings force the specified information transfer capability to be used except for the transparent setting, which indicates that the information transfer capability of the call should be forwarded transparently.

Example: Configure information transfer capability handling

In the following example the information transfer capability for inbound calls through the H.323 interface *Myh323If* is forced to speech. This is an appropriate setting, when communicating to Microsoft Netmeeting clients.

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface h323 MyH323If
node(if-h323)[MyH323If]#itc rx speech
```

Configuring CLIP/CLIR support (optional)

According to the H.323 standard, information about *calling line identification presentation/calling line identification restriction* (CLIP/CLIR) is not provided, when using the H.323 protocol. However, there are H.323 equipment vendors, which allow tunneling this information through an H.323 connection. The additional information is inserted in octed 3a of the calling party number information element in the Q.931 part of the H.323 setup message.

Note This functionality is not standardized and might cause interoperability problems, if enabled.

Procedure: To enable tunnelling of CLIP/CLIR information over H.323

Mode: Interface H.323

Step	Command	Purpose
1	***node*(*if- h323*)[*if-name*]#clip-clir-support**	Enables CLIP/CLIR support on the H.323 interface

Example: Enable CLIP/CLIR support

The following example shows how to enable CLIP/CLIR support on the H.323 interface MyH323If.

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface h323 MyH323If
node(if-h323)[MyH323If]#clip-clir-support
```

Enabling 'early-proceeding' on H.323 interfaces

The **early-proceeding** command can enable the early-proceeding feature on H.323. If this feature is enabled, the gateway will immediately reply with an *H.225* (H.323) call-proceeding message in response to a received *H.225* (H.323) setup message without waiting for a response from the destination.

Mode: interface h323 <if-h323>

Step	Command	Purpose
1	**[***name***](if-h323)[***if-name***]#early-proceeding**	Enables the early-proceeding feature.

Example: Enable early call disconnect

The following example shows how to enable early call disconnect on an H.323 interface named MyH323If.

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface h323 MyH323If
node(if-h323)[MyH323If]#early-disconnect
```

Enabling the early call connect (optional)

The **early-connect** command can be enabled to open a data path before the call is connected to play inband information from the ISDN side.

Mode: interface h323

Step	Command	Purpose
1	**[***name***](if-h323)[***name***]#early-connect**	Enables the early-connect feature.

Enabling the early call disconnect (optional)

Early call disconnect suppresses busy tones (e.g. disturbing a telephone conference) and post-call announcements by sending an H.323 Release message to the remote peer when the connected terminal hangs up (ISDN: when Disconnect message is received; analog line: when busy tone is detected, loop current is interrupted, or battery voltage is reversed).

Procedure: To enable early call disconnect

Mode: Interface H.323

Step	Command	Purpose
1	***node*(if-h323)[*if-name*]#early-disconnect**	Enables early call disconnect (Default: disabled)

Example: Enable early call disconnect

The following example shows how to enable early call disconnect on an H.323 interface named MyH323If.

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface h323 MyH323If
node(if-h323)[MyH323If]#early-disconnect
```

Enabling the via address support (optional)

Some LAN Voice applications require the H.323 gateway to add the calling party number of the connected terminal as an H.323 E.164 Alias to the Facility message when transferring a call to another gateway. This enables a gatekeeper to detect loops of call forwarding and to stop them.

Procedure: To enable sending of the via address in call transfers

Mode: Interface H.323

Step	Command	Purpose
1	***node*(if-h323)[*if-name*]#via-address-support**	Enables sending of the via address in call transfers (Default: disabled)

Example: Enabling the via address support

The following example shows how to enable the via address support on an H.323 interface named MyH323If.

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface h323 MyH323If
node(if-h323)[MyH323If]#via-address-support
```

Override the default destination call signaling port (Optional)

Normally, if no gatekeeper is used, the TCP call-signaling connection for outbound H.323 calls is established to the H.323 standard call-signaling port 1720. If your destination uses a different call-signaling port, it is possible to define an alternate port using this procedure.

Note The call-signaling port specified here has no effect, if a gatekeeper is used. In that case the gatekeeper will provide the portnumber to use for establishing the call signaling connection

Procedure: To configure an alternate destination TCP call-signaling port

Mode: Interface H.323

Step	Command	Purpose
1	***node*(*if-h323*)[*if-name*]# remoteport** *port*	Specifies the TCP port to which the call-signaling connection should be established.

Example: Specifying an alternate destination call-signaling port

The following example shows how to set the destination call-signaling port number for the H.323 interface *MyH323If* to 2300.

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface h323 MyH323If
node(if-h323)[MyH323If]#remoteport 2300
```

Configuring status inquiry settings (optional)

Normally, the H.323 gateway will send out status inquiries every minute on each connected H.323 call. According to the H.323 standard, the remote entity must respond to these status inquiries, which allows the H.323 gateway to detect, if the call on the remote H.323 entity is still alive. If no response is received after another minute, the call will be dropped.

Unfortunately, there are H.323 entities, which do no respond to these status inquiries. This causes every call to be dropped after being connected for two minutes using the default setting.

As a workaround for these non-compliant implementations, it is possible to disable the status inquiry checking.

It is also possible to change the default status inquiry interval of 60 seconds to a different value, if required.

Procedure: To disable status inquiries

Mode: Interface H.323

Step	Command	Purpose
1	***node*(*if-h323*)[*if-name*]#no status-inquiry**	Disables status inquiries on the interface

Example: Disable status inquiries

The following example shows how to disable status inquiries for calls handled by the H.323 interface *MyH323If*.

```
node>enable
node#configure
```

```
node(cfg)#context cs
node(ctx-cs)[switch]#interface h323 MyH323If
node(if-h323)[MyH323If]#no status-inquiry
```

Procedure: To change the default status inquiry interval

Mode: Interface H.323

Step	Command	Purpose
1	***node*(*if-h323*)[*if-name*]#status-inquiry timeout** seconds	Changes the status inquiry interval on the interface to the specified number of seconds

Example: Disable status inquiries

The following example
the status inquiry interval for the H.323 interface *MyH323If* to 120 seconds.

```
node>enable
node#configure
node(cfg)#context cs
node(ctx-cs)[switch]#interface h323 MyH323If
node(if-h323)[MyH323If]#status-inquiry timeout 120
```

AOC-D Support for H.323

The H.323 gateway is able to accept and send Advice of Charge during the call (AOC-D) messages according to the ITU-T standard Q.956. Facility Information Elements (IEs) in the Q.931 portion of the protocol are used to transport AOC-D PDUs. (Refer to “Chapter 28: ISDN interface configuration”, section “ISDN Advice of Charge support.”)

You can enable/disable reception and transmission of AOC-D messages separately on each H.323 interface.

When reception is enabled, AOC-D messages received in incoming H.323 FACILITY messages are forwarded to the ISDN side of the call. In addition to the H.323 interface, AOC support must also be enabled on the ISDN interface (see ISDN Advice Of Charge support section of Chapter 28 ISDN interface configuration of the Software Configuration Guide).

When transmission is enabled, AOC-D messages received from the ISDN side of the call are sent as H.323 FACILITY to the remote terminal or gatekeeper.

The following commands can be used to change the AOC-D over H.323 tunneling behavior on an H.323 interface:

Mode: context cs/interfce h.323 <interface-name.

Step	Command	Purpose
1	**node(if-h323)[if-name]# [no] aoc-d accept**	Enables or disables reception of AOC-D information in FACILITY messages received in calls over the current interface. Default: Reception is disabled. When enabled, Advice of Charge information is received and accepted from the other side of the gateway.
2	**node(if-h323)[if-name]# [no] aoc-d emit**	Enables or disables transmission of AOC-D information in FACILITY messages received from the other side of the gateway. Default: Transmission is disabled. When enabled, the H.323 gateway sends FACILITY messages containing AOC-D PDUs whenever the charge of a call changes.

Chapter 39 SIP interface configuration

Chapter contents

Introduction 442
SIP interface configuration task list 442
Binding the interface to a SIP gateway 443
Configure a remote host 443
Configuring a local host (Optional) 444
Using an alternate VoIP profile (Optional) 444
Using an alternate SIP profile (Optional) 445
Using an alternate Tone-Set profile (Optional) 445
Configuring early call connect / disconnect (Optional) 446
Configuring address translation (Optional) 446
Mapping call-control properties in SIP headers 446
Mapping SIP headers to call-control properties 447
Configuring ISDN Redirecting Number Tunneling Over SIP 447
Enabling SIP RFC Privacy, Asserted-Identity, & Preferred-Identity headers (RFC 3323/3325) 448
Updating caller address parameters 448
SIP Diversion Header 449
Transmit Direction 450
Receive Direction 450
SIP REFER Transmission (& ISDN Explicit Call Transfer support) 451
AOC Over SIP (Optional) 453
Enabling the session timer (Optional) 454
Enabling the SIP penalty-box feature (Optional) 454
Initiating a new SIP session for redirected SIP calls (Optional) 454
Configure the SIP hold method (Optional) 455

Introduction

This chapter provides an overview of SIP interfaces used by context SIP gateways and describes the specific tasks involved in their configuration. This chapter does not explain the basic configuration steps required for all CS interfaces. See Chapter 33, "CS interface configuration" on page 381 for information about configuring basic CS interfaces.

Within the CS context, a SIP interface is a special type of CS interface providing call routing for incoming and outgoing calls to and from the context SIP gateway (see Figure 60).

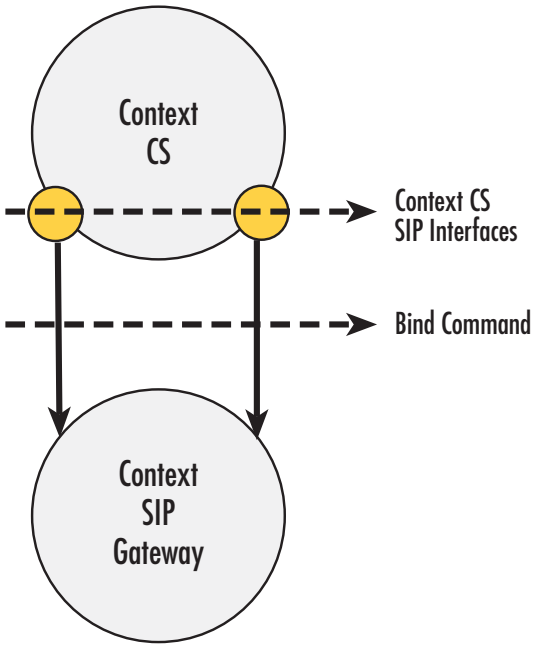


Figure 60. SIP interfaces on the CS context

A SIP interface is a CS interface type that is responsible for the address translation between the SIP environment and the call control. It provides configurable translation rules that define which parameter of which SIP header has to be translated to call control properties and vice versa. In addition, it offers VoIP settings and the possibility to configure SIP supplementary services like Call Transfer, Call Reroute or Session Timer. All SIP interfaces must be explicitly bound to a context SIP gateway. Calls that are routed from the Context CS to one of the SIP interfaces will be forwarded for call establishment to the context SIP gateway to which the SIP interface is bound. All of the parameters configured in the SIP interface will be applied to the forwarded call.

SIP interface configuration task list

This section describes the configuration tasks for SIP interfaces listed below. You must at least perform the tasks that are not marked as optional to define a working SIP interface. The optional tasks are only required in advanced configurations. Before you can start with the SIP interface specific configuration tasks, you need to create the SIP interface and define the routing for it as defined in Chapter 33, "CS interface configuration" on page 381.

- Binding the interface to a context SIP gateway (see page 443)
- Configure a remote host (see page 443)
- Configure a local host (Optional) (see page 444)
- Using an alternate VoIP profile (Optional) (see page 444)
- Using an alternate SIP profile (Optional) (see page 445)
- Using an alternate Tone-Set profile (Optional) (see page 445)
- Configuring early call connect /disconnect (Optional) (see page 446)
- Configuring address translation (Optional) (see page 446)
- SIP REFER Transmission (& ISDN Explicit Call Transfer support) (Optional) (see page 451)
- AOC over SIP (Optional) (see page 453)
- Enable the session timer (Optional) (see page 454)
- Enable the SIP penalty-box feature (Optional) (see page 454)
- Initiating a new SIP session for redirected SIP calls (Optional) (see page 454)
- Configure the SIP hold method (Optional) (see page 455)

Binding the interface to a SIP gateway

Every SIP interface must be explicitly bound to a context SIP gateway. It defines the transport interfaces to be used to send requests and also from where it expects to receive incoming calls.

Mode: Interface SIP

Step	Command	Purpose
1	[*node*]**(if-sip)[if-name]# [no] bind context sip-gateway** *gw-name*	Binds the interface to a context SIP gateway. The no form of the command removes an existing binding. *Default:* none

Configure a remote host

The remote host parameter is used to build the host part of the To-Header-URI and the Request-URI. All calls forwarded to the context SIP gateway through this SIP interface will be sent to that host unless an outbound proxy has been configured for the outgoing Request-URI. In this case, all requests will be sent to the specified proxy independent of the Request-URI. For details about proxy configuration, see Chapter 51, "Location Service" on page 607.

The remote host parameter may contain a Domain Name, a Full Qualified Domain Name or an IP Address. If domain names are used, do not forget to configure the DNS client for address resolving. See Chapter 28, "DNS configuration" on page 304 for more information.

An optional port number can be entered on this command. It specifies the destination port of the requests to be sent. If no port has been defined, the default SIP signaling port 5060 will be taken.

Mode: Interface SIP

Step	Command	Purpose
1	[*node*](**if-sip**)[**if-name**]# [**no**] **remote** <*host-name*> [**port**]	Specifies the remote host name and port. The no form of the command removes a configured host name. *Default Hostname:* none *Default Port:* 5060

Configuring a local host (Optional)

The local host parameter is used to build the host part of the From-Header-URI. Optionally, a port can be entered. If no port has been specified, it will not appear in the From-Header-URI. This command is optional in a SIP network setup where no DNS names are involved and only IP addresses applied. If no local host name has been specified, the IP address of the outgoing IP interface will be taken as host part of the From-Header-URI.

There are some exceptions where a local host name must be entered. One exception is if the bound gateway has two transport interfaces and the From-Header-URI call outbound properties have been configured. Such a call outbound property is the outbound proxy. To find the right identity in the location service, the From-Header-URI must exist. But, if there is more than one transport interface, it is not clear which one will be used to send the request because the proxy influences the routing. In such a case, it is recommended that the user configures the local host name and knows which IP interface the request should be sent over.

The SIP server expects its own IP address in the From-Header-URI because of security or routing reasons. This can be solved by manually entering a local host name.

Mode: Interface SIP

Step	Command	Purpose
1	[*node*](**if-sip**)[**if-name**]# [**no**] **local** <*host-name*> [**port**]	Specifies the local host name and port. The no form of the command removes a configured host name. *Default Hostname:* none *Default Port:* none

Using an alternate VoIP profile (Optional)

A VoIP profile is a container for all datapath-related settings on VoIP connections. The predefined default profile exists persistently in the system and it is preconfigured with proper default parameters. It will always be taken if no other profile has been specified. This command allows the user to specify an alternate VoIP profile to use for all calls routed over this interface. For details about VoIP profile configuration see Chapter 47, "VoIP profile configuration" **on page 573**.

When a profile has been specified on the interface, it is possible to provide a special one for specific identities or group of identities. It can be configured in location service identities for call inbound and call outbound. See Chapter 51, "Location Service" **on page 607** for details about identities and VoIP profile configurations. The identity lookup to find a possible configured VoIP profile will always be applied to the remote SIP URI. For outgoing calls, the SIP Request-URI will be taken. For incoming calls, the SIP From-URI will be taken.

Mode: Interface SIP

Step	Command	Purpose
1	**[***node***](if-sip)[if-name]#use profile voip** *profile-name*	Defines an alternate VoIP profile to be used for this SIP interface. *Default:* default

Using an alternate SIP profile (Optional)

The SIP profile contains a cause and reason mapping from SIP notation to the call control notation and vice versa. The predefined default profile exists persistently in the system and it is preconfigured with proper default mappings. It will always be taken if no other profile has been specified. This command allows the user to specify an alternate SIP profile to use for all calls routed over this interface. For details about SIP profile configuration see Chapter 49, "SIP profile configuration" **on page 601**.

When a profile has been specified on the interface, it is possible to provide a special one for specific identities or group of identities. It can be configured in location service identities for call inbound and call outbound. See Chapter 51, "Location Service" **on page 607** for details about identities and SIP profile configurations. The identity lookup to find a possible configured SIP profile will always be applied to the remote SIP URI. For outgoing calls, the SIP Request-URI will be taken. For incoming calls, the SIP From-URI will be taken.

Mode: Interface SIP

Step	Command	Purpose
1	**[***node***](if-sip)[if-name]#use profile sip** *profile-name*	Defines an alternate SIP profile to be used for this SIP interface. *Default:* default

Using an alternate Tone-Set profile (Optional)

A Tone-Set profile contains a mapping of call-progress tones to defined tone sequences. The predefined default profile exists persistently in the system and it is preconfigured with the Swiss standard call-progress tone mapping. It will always be taken if no other profile has been specified. This command allows the user to specify an alternate Tone-Set profile to be use for all calls routed over this interface. For details about Tone-Set profile configuration see Chapter 42, "Tone configuration" **on page 529**.

When a profile has been specified on the interface, it is possible to provide a special one for specific identities or group of identities. It can be configured in location service identities for call inbound and call outbound. See Chapter 51, "Location Service" **on page 607** for details about identites and Tone-Set profile configurations. The identity lookup to find a possible configured Tone-Set profile will always be applied to the remote SIP URI. For outgoing calls, the SIP Request-URI will be taken. For incoming calls, the SIP From-URI will be taken.

Mode: Interface SIP

Step	Command	Purpose
1	[*node*]**(if-sip)[if-name]#use profile tone-set** *profile-name*	Defines an alternate Tone-Set profile to be used for this SIP interface. *Default:* default

Configuring early call connect / disconnect (Optional)

Normally, SIP calls are fully connected by sending a 200 OK response to the INVITE request, if the called party answers the call. Any call progress tones or announcements are transmitted in the early SIP dialog. However, there are several SIP user agents that do not support media to be transmitted or received in an early dialog. To solve this problem, it is possible to connect the SIP call using a 200 OK response to the initial INVITE request as soon as in-band information is available. This will allow any SIP user agent to receive pre-call inband information.

Early call disconnect suppresses busy tones (e.g. disturbing a telephone conference) and post-call announcements by sending a BYE message to the remote SIP user agent when the connected terminal hangs up (ISDN: when Disconnect message is received; analog line: when busy tone is detected, loop current is interrupted, or battery voltage is reversed).

Mode: Interface SIP

Step	Command	Purpose
1	[*node*]**(if-sip)[if-name]#[no] early-connect**	Enables/Disables early call connect *Default:* disabled
2	[*node*]**(if-sip)[if-name]#[no] early-disconnect**	Enables/Disables early call disconnect *Default:* disabled

Configuring address translation (Optional)

Mapping call-control properties in SIP headers

This functionality specifies rules for building the SIP headers for outgoing SIP requests. The user can configure which call-control property should take place as the user-part, the host-part or as additional parameter in a SIP header's URI.

Mode: Interface SIP

Step	Command	Purpose
1	[*node*]**(if-sip)[if-name]#[no] address-translation outgoing-call** *{header} {parameter} {source} {property}*	Specifies an outgoing address translation rule.

Mapping SIP headers to call-control properties

This functionality specifies rules that describe which SIP header should take place as a specific call-control property like the called-e164 or the calling-e164 number.

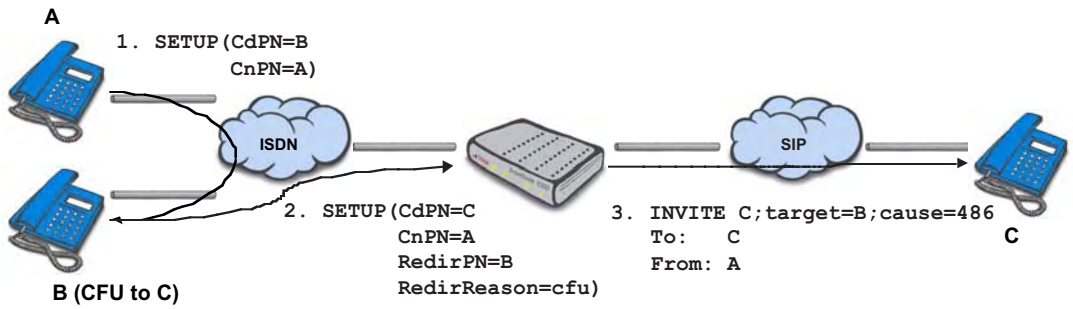
Mode: Interface SIP

Step	Command	Purpose
1	[*node*]**(if-sip)[if-name]#[no] address-translation incoming-call** *{property} {header}*	Specifies an incoming address translation rule.

Configuring ISDN Redirecting Number Tunneling Over SIP

A SIP interface can be configured to tunnel the ISDN Redirecting E.164 Number and Redirecting Reason. This is implemented per the IETF draft standard method according to draft-jennings-sip-voicemail-uri-05.

The figure below shows a call that is redirected in the PSTN network and reaches the device over an ISDN interface. The incoming ISDN SETUP message contains additional information elements for the Redirecting Party Number (B) and the Redirecting Reason (Call-Forwarding-Unconditional, for example). When the Redirecting Number Tunneling over SIP has been enabled, the Request-URI of the outgoing SIP INVITE message has additional parameters for the redirected number (target) and the redirecting reason (cause).



Normally, the Redirecting Number Tunneling over SIP is disabled. However, you can manually enable this feature for each SIP interface separately using the following commands. Note that transmission and reception of the target and cause parameters must be configured separately.

Mode: Interface SIP

Step	Command	Purpose
1	[*node*]**(if-sip)[if-name]# address-translation outgoing-call request-uri target-param call redir**	Enables Redirecting Party Number Tunneling ISDN ‡ SIP: Enables transmission of the target and cause parameters in the Request-URI for outgoing SIP calls. Whenever the (incoming ISDN) call has a redirecting party number information element, the Request-URI is extended by the target and cause parameters. *Default:* disabled
2	[*node*]**(if-sip)[if-name]# address-translation incoming-call calling-redir requesturi-target-param**	Enables Redirecting Party Number Tunneling SIP ‡ ISDN: Enables reception of the target and cause parameters in the Request-URI for incoming SIP calls. Sets the redirecting party number information element for (outgoing ISDN) calls where the target parameter is present in the Request-URI. *Default:* disabled

Enabling SIP RFC Privacy, Asserted-Identity, & Preferred-Identity headers (RFC 3323/3325)

The following command sequence enables support for the SIP Privacy and Asserted-Identity headers, sending and receiving of the Privacy as well as the Asserted-Identity headers of RFCs 3323 & 3325. This provides the required identity to SIP entities. The privacy header suppresses the forwarding of the identity to the final station. Handling of the Asserted-Identity header can be configured in the same way as for any other SIP header using the address-translation command in the SIP interface configuration mode. The privacy header is mapped to the call-control's presentation indicator property field.

The type of header sent depends on the screening-indicator property of the call-control. When receiving one of these privacy headers the screening-indicator is also set depending on the received header type.

Mode: Interface SIP

Step	Command	Purpose
1	[*node*]**(if-sip)[if-name]#[no] privacy**	Enables/Disables privacy. *Default:* disabled

Updating caller address parameters

A SIP User Agent Client (UAC) can use the sip update method for modifying caller-name and caller-number. These parameters must be included in the P-Asserted-Identity header of the sip update message. This feature is often required if sip has to interwork with an analog telephone network where the Caller-Id cannot be delivered together with the call offering. If SmartWare is working as SIP to FXS Gateway, the UAC can send the update at any time between the initial invite and the final response. The delivered parameters will be presented to the FXS subscriber during the current or the next ring pause.

It is possible to configure the sip interface to wait for the caller-name, caller-number or both parameters before routing the call. If the caller-address parameters are sent with sip update and the call will be forwarded to a network where these parameters must be present at call setup time (ISDN), this wait-service must be enabled.

Note The SIP interface of the call-router must be configured to take the caller addresses parameters for incoming calls from the P-Asserted-Identity header. See "Enabling SIP RFC Privacy, Asserted-Identity, & Preferred-Identity headers (RFC 3323/3325)" on page 448 for more information.

Mode: Interface SIP

Step	Command	Purpose
1	[*node*]**(if-sip)**[*interface*]**#[no] update accept address {wait-for-name | wait-for-number | wait-for-name-and-number} [[proceeding-timeout** *<timeout>*]**[early-alerting timeout** *<timeout>*]]	Enables or disables wait service. Wait Options: • **wait-for-name** The call will be routed as soon as a valid caller name is present. • **wait-for-number** The call will be routed as soon as a valid caller number is present. • **wait-for-name-and-number** The call will be routed as soon as a valid caller name and a valid caller number are present. Wait Timeouts: • **proceeding-timeout** After this time, the call will be routed and the specified caller parameters are not present. *Default:* 4000ms • **early-alerting-timeout** If the specified caller parameters are not present after this time, a 180 RINGING will be sent to UAC. *Default:* 0ms (immediately)

SIP Diversion Header

SmartWare supports the SIP Diversion Header for transmitting redirecting information over SIP according to draft-levy-sip-diversion-08. Sending and receiving of the header can be configured independent of each other. Even though the Diversion Header standard would allow appending a header for each diversion occurred in the network, SmartWare only records the last and the first diversion. If only one Diversion Header is attached to the INVITE request, then it represents the last diversion.

Transmit Direction. For enabling sending of the Diversion Header, an outgoing address translation expression must be configured on the sip interface. This expression specifies how to create the Diversion URI of the header. As the User Part of the URI, the Calling Redirecting number will always be taken. The user must configure the Host Part that is set per default to none. Setting the Host Part to none disables transmission of the Diversion Header.

Mode: Interface SIP

Step	Command	Purpose
1	**[name] (if-sip)[*interface*]#address-translation outgoing-call diversion-header host-part {call | default-server | domain | fix | interface | none}**	Enables or disables sending of the Diversion Header and specifies the Host Part of the URI. • **call:** If available, the Host Part of the calling from-header will be taken else the local ip address. • **default-server:** The ip address of the configured default-server will be taken. • **domain:** The configured domain name will be taken. • **fix:** Allows to specify a user configured Host Part. • **interface:** The local ip address will be taken. • **none:** Disables sending of the Diversion Header.

Receive Direction. For receiving of the Diversion Header, an incoming address translation expression must be configured on the sip interface. Because several methods for transmitting redirecting information are available, this expression specifies that they must be taken from the Diversion Header when providing them to the call control.

Mode: Interface SIP

Step	Command	Purpose
1	**[name] (if-sip)[*interface*]#address-translation incoming-call calling-redir diversion-header**	Enables or disables extracting of the redirection information from the Diversion Header.

SIP REFER Transmission (& ISDN Explicit Call Transfer support)

Additional call transfer support, a push-back mechanism, is enabled by default for SIP interfaces by sending REFER messages.

SmartWare detects calls that are looped internally, i.e. calls that leave the device over the same SIP interface over which they enter the device. If an internal loop is detected for a SIP interface, SmartWare sends a REFER message to push back the call to the connected network as soon as the call is connected.

ISDN interfaces react similarly to internally looped calls. The push-back mechanism is enabled by default for ISDN interfaces (BRI ports) by accepting or rejecting explicit call-transfer (ECT) invocations. An ISDN phone that is connected to a BRI port and that has two active calls can send an ECT invocation to connect the two calls inside the device. An ISDN interface can be configured to accept or reject ECT invocations.

SmartWare detects calls that are looped internally, i.e. calls that leave the device over the same ISDN interface over which they enter the device. If an internal loop is detected for an ISDN interface bound by an ISDN user port, SmartWare sends an explicit call-transfer (ECT) to push back the call to the connected network as soon as the call is connected. An ISDN interface can be configured to emit ECT invocations.

Figure 61 on page 452 shows an example scenario where a SIP network connects two devices to give a home office access to a PBX in the central office.

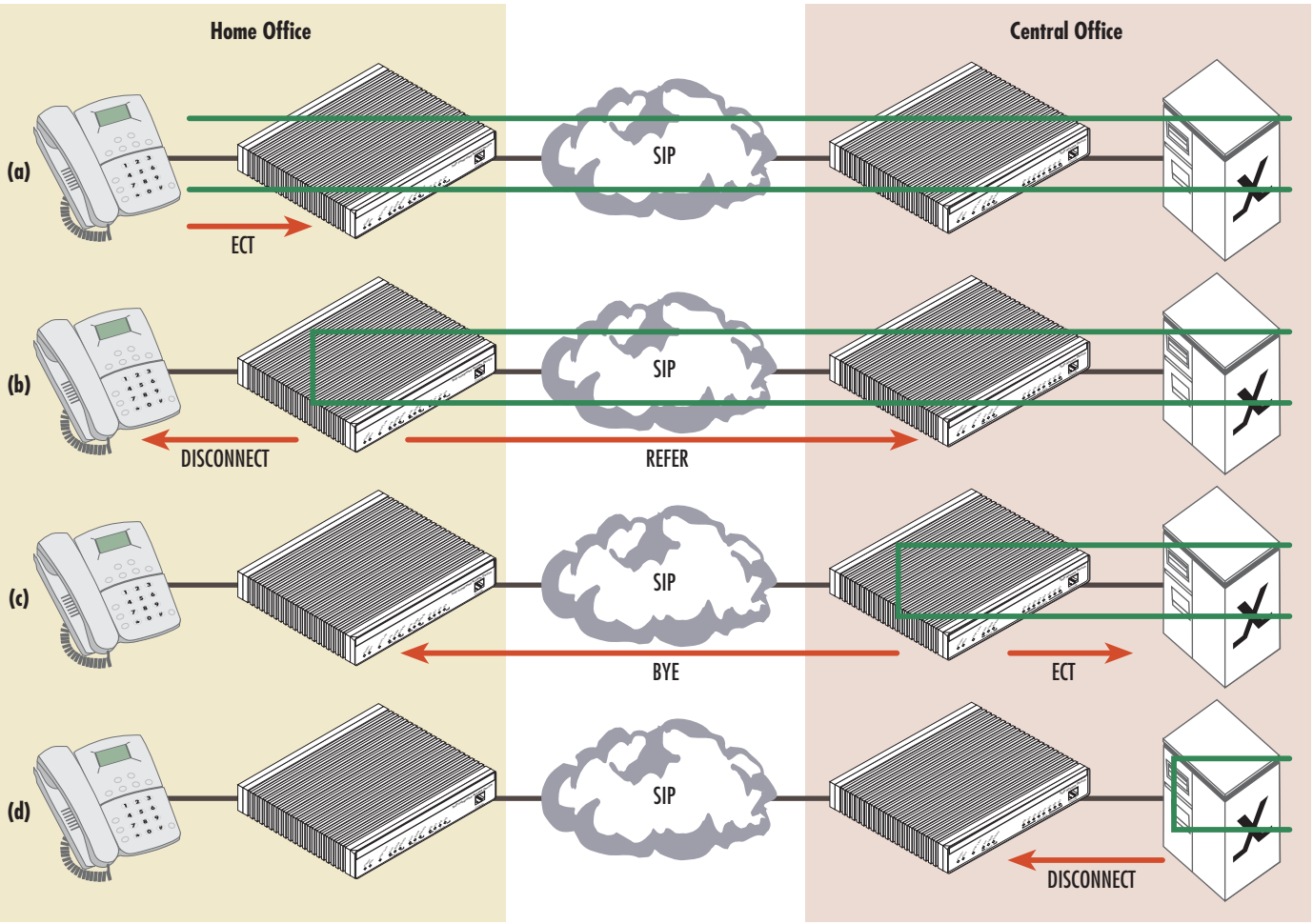


Figure 61. Example SIP network connecting two devices to give a home office access to the CO PBX

The phone in the home office has two active calls to other subscribers of the PBX in the central office. The user wants to connect the other two participants and (a) sends an explicit call-transfer invocation to the device HO. The device HO internally connects the two calls and sends a DISCONNECT message to the phone for both calls. In a second step (b) the firmware on HO detects an internal loop. Both call legs are connected to the same network. In this example, both call legs are handled by the same SIP gateway. The firmware on device HO sends a REFER message to device CO, which connects the two call legs internally and sends a BYE message to the device HO. (c) Again the firmware of CO detect an internal loop. This time the call legs are handled by the same SIP interface, connected to the PBX. Since the ISDN port is a user port it sends an explicit call-transfer invocation to the PBX (d), which connects the call and sends the device CO a DISCONNECT message for both calls. During all these push back operations the datapath of the two participants keeps connected.

The push back mechanism over ISDN (using ECT) and SIP (using REFER) works independently of the protocol that invoked the call-transfer. For example, the same scenario also works if the phone in the home office is connected to an FXS port.

The push-back mechanism can be configured on each interface separately. Per default push-back is enabled for ISDN and SIP interfaces. You only have to change the configuration if you don't want internally looped calls to be pushed back to the network. The configuration command [**no**] **call-transfer accept** configures if an incoming call-transfer request (e.g. ECT or REFER) shall be accepted. The configuration command [**no**] **call-transfer emit** configures if a call reaching the device over this interface and leaving the device over this interface shall be pushed back to the network, i.e. if a call-transfer request (ECT or REFER) shall be sent.

The following procedure disables the push-back mechanism on the ISDN interface connected to the PBX. No ECT invocation is sent when a call is detected that is looped internally.

Mode: Interface ISDN

Step	Command	Purpose
1	[*node*]**(if-isdn)[if-name]#no calll-transfer emit**	Disables the ISDN push back mechanism.

The following procedure disables the push-back mechanism on a SIP interface. No REFER message is sent when a call is detected that is looped internally.

Mode: Interface SIP

Step	Command	Purpose
1	[*node*]**(if-sip)[if-name]#no calll-transfer emit**	Disables the SIP push back mechanism.

AOC Over SIP (Optional)

This enhancement sends AOC information transparently from ISDN (or H.323) to SIP and vice versa. AOC-D elements are hex-encoded and sent as application/QSIG content in SIP INFO messages during a session.

Mode: Interface SIP

Step	Command	Purpose
1	**[name] (if-sip)[***interface***]#[no] aoc-d accept**	Enables or disables the reception of SIP Info messages containing AOC-D elements and propagate charging information to adjacent peer.
2	**[name] (if-sip)[***interface***]#[no] aoc-d emit**	Enables or disables the sending of SIP Info messages with AOC-D elements containing charging information from adjacent peer. If no charging information is available, no message is sent.

Enabling the session timer (Optional)

The gateway implements the SIP session timer feature, which is currently only defined in SIP draft standards. The session timer feature allows a gateway to check periodically during a call, if the remote gateway is still alive and if the call is still connected on the remote gateway. You can enable this feature using the command shown below. If the session timer feature detects that the call is no longer connected on the remote side, it will also drop the call locally.

Mode: Interface SIP

Step	Command	Purpose
1	[*node*]**(if-sip)[if-name]#[no] session-timer seconds**	Enables/Disables the session timer with a refresh period of the specified number of seconds. *Default:* disabled

Enabling the SIP penalty-box feature (Optional)

The following command enables the SIP penalty-box feature, which causes a non-responsive server not to be contacted again for the specified duration. A server will be in the penalty box for the specified time as soon as a single transaction with that server fails.

Mode: Interface SIP

Step	Command	Purpose
1	[*node*]**(if-sip)[if-name]#[no] penalty-box**	Enables/Disables the SIP penalty box feature. *Default:* disabled

Initiating a new SIP session for redirected SIP calls (Optional)

Normally, if a SIP call is redirected to a different location by receiving a SIP 3xx response, only the request header is replaced in the original INVITE message and the INVITE is sent to the new destination. Using the following procedure, the SIP gateway can be forced to create a completely new session for forwarded calls.

Mode: Interface SIP

Step	Command	Purpose
1	[*node*]**(if-sip)[if-name]#[no] new-session-after-redirect**	Enables/disables creation of a new session after a redirect response. *Default:* disabled

Configure the SIP hold method (Optional)

There are different ways to set a remote SIP subscriber On Hold. This command specifies which method the SmartNode uses to indicate this call state. In receive direction, all of them will be accepted.

Mode: Interface SIP

Step	Command	Purpose
1	[*node*]**(if-sip)[if-name]#hold-method {direction-attribute | zero-ip}**	Configures the method to be used to signal the hold state. *Default:* zero-ip

Chapter 40 Call router configuration

Chapter contents

Introduction 458
Call router configuration task list 460
Map out the goals for the call router 460
Enable advanced call routing on circuit interfaces 461
Configure general call router behavior 461
Configure address completion timeout 461
Configure default digit collection timeout and terminating character 462
Configure number prefix for ISDN number types 463
Configure call routing tables 464
Create a routing table 464
Called party number routing table 466
Regular Expressions 466
Digit Collection 468
Digit Collection Variants 469
Calling party number routing table 472
Number type routing table 472
Numbering plan routing table 473
Name routing table 474
IP address routing table 474
URI routing table 475
Presentation Indicator Routing Table 475
Screening Indicator Routing Table 476
Information transfer capability routing table 477
Call-router support for redirecting number and redirect reason 478
Time of day routing table 479
Day of Week Routing Table 479
Date routing table 479
Deleting routing tables 480
Configure mapping tables 481
E.164 to E.164 Mapping Tables 485
Custom SIP URIs from called-/calling-e164 properties 488
Other mapping tables 488
Deleting mapping tables 489
Creating complex functions 490
Deleting complex functions 491
Digit collection & sending-complete behavior 492
Sending-Complete 492
Ingress interface 492
Call-Router 493

Egress Interface 495
Creating call services 497
Creating a hunt group service 497
Creating a distribution group service 506
Distribution-Group Min-Concurrent setting 508
Call-router 'limiter' service 508
Priority service 509
CS Bridge service—'VoIP Leased Line' 511
Configuring the service second-dialtone 513
Deleting call services 514
Activate the call router configuration 514
Test the call router configuration 515
Configure partial rerouting 521
Call reroute 522
Enable acceptation of rerouting requests on ISDN. 522
Enable emission of rerouting requests on ISDN. 522
Enable sending of "302 moved temporary" message on SIP. 522
Allow Push-Back 522
Enable push-back – aaa service 522
Enable push-back – bridge service 523
Enable push-back – distribution-group service 523
Enable push-back – hunt group service 523
Enable push-back – limiter service 523
Enable push-back – priority service 523

Introduction

This chapter provides an overview of call router tables, mapping tables and call services and describes the tasks involved in configuring the call router in SmartWare. This chapter includes the following sections:

- Call router configuration task list
- Call router configuration tasks
- Examples

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

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

The call router is a very efficient and flexible tool for routing calls between CS interfaces. Based on a set of routing criteria, the call router determines the destination (interface) for every incoming call. The forwarding decisions and features are based on a set of routing tables, mapping-tables and call services.

Each routing table is responsible for a specific routing criterion such as the called party number or the bearer capability of the call. Multiple tables can be linked together to form a decision tree. The mapping tables can be used to modify call properties like the calling and called party numbers according to the network requirements. Call services can be used in the routing path to observe the call state and spawn other calls. The hunt-group

service is an example for a call service. Figure 62 illustrates direct call and advanced call routing. In this chapter, advanced call routing is explained. For configuring direct call routing refer to chapter 40, "Call router configuration" on page 456.

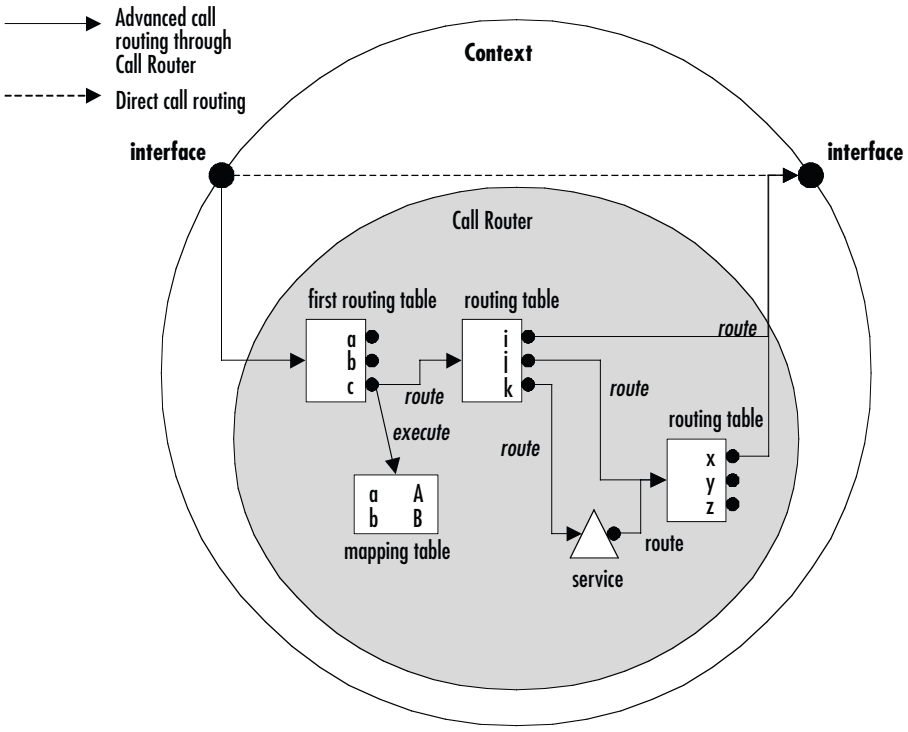


Figure 62. Direct call routing vs. advanced call routing

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

- Calling party number (*calling-e164*); also called *source-Nr*, *A-Nr*, *MSN*, *DDI*, or *CLIP*
- Called party number (*called-e164*); also called *destination-Nr* or *B-Nr*
- Calling and called party number type
- Calling and called party number plan
- Calling and called party name, the display name
- Calling and called party IP address (for VoIP calls)
- Calling and called party URIs (for SIP calls)
- Presentation indicator; whether the number shall be presented to the other party
- Screening indicator; whether the number has been screened
- Information transfer capability; also called ISDN bearer capability or ISDN service
- Day of week; Monday–Sunday

- Time of day; *hour:minute:second*
- Date; *day.month.year*

The call router allows you to solve practically any call routing and call property manipulation requirement that you may have. The call router is very flexible in allowing the construction of decision trees based on linked routing tables. However you should take care not to use too many tables and an over-elaborate structure. The configuration may become large and difficult to manage. For complex configurations we recommend offline editing and configuration downloads.

Call router configuration task list

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

- Map out the goals for the call router
- Enable advanced call routing on circuit interfaces
- Configure general call router behavior
- Configure call router tables
- Configure mapping tables and complex functions
- Configure call services
- Deleting routing tables and functions
- Activate the call router configuration
- Test the call router configuration

Map out the goals for the call router

There are many possible policies and factors that may influence the call router configuration. Some examples are:

- On-net off-net call routing
- ISDN service routing
- Carrier selection
- Service quality
- Fallback strategies
- Network and gateway selection

Other factors that must be taken into account are:

- Available number ranges (DDI, MSN, PISN)
- Potential restrictions imposed by neighboring equipment (Gatekeepers, Remote Gateways, PBXs) on the number length or range to be used.

The call router is able to accommodate almost every combination of these requirements through a customized configuration.

In order to keep this configuration compact we recommend that you first define the routing requirements and restrictions that apply to your installation. Then define the routing and mapping tables and the call services that you need to fulfill these requirements. Finally define the decision tree (i.e. the sequence in which the tables are linked together). In this step you may realize that you need multiple tables of the same type to achieve your goals. On the other hand an alternative sequence may help you to reduce the number of tables or the size of each table while still achieving the set goal. Only when you are happy with the planned tables, functions and sequence should you start configuration.

Enable advanced call routing on circuit interfaces

To activate the advanced call routing the first routing table in the CS interface has to be specified as you can see in figure 62. Make sure the route parameter on the CS interface is configured in order to forward the incoming calls to the first table of the call router.

Procedure: To enable advanced call routing on a circuit interface

Mode: Context CS

Step	Command	Purpose
1	node(ctx-cs)[switch]#**interface** *if-type if-name*	Change to Interface Configuration Mode to specify the entry table in the interface
2	*node*(if-*type*)[*if-name*]#**route call dest-interface** *interface-name* *or* *node*(if-*type*)[*if-name*]#**route call dest-table** *table-name* *or* *node*(if-*type*)[*if-name*]#**route call dest-service** *service-name*	Use these commands to route a call (1) directly to an interface specified with the *interface-name* parameter; (2) to the call router, using the *table-name* table as first routing table; (3) directly to a service specified with the *service-name* parameter.

Configure general call router behavior

Configure address completion timeout

A call that is routed through a called party number routing table possibly has a called party number that is too short for a routing decision to be made. In this case the call router waits for additional digits being entered by the calling user. When the user does not enter additional digits during the address completion timeout, the call router drops the call. You can configure the address completion timeout following the procedure below. The default address completion timeout is 12 seconds and is restarted after each digit that is sent during overlap dialing.

> **Note** The address completion timeout is active when the call router waits for mandatory digits before being able to complete call routing. Contrary to this, the digit collection timeout, described below, waits for additional optional digits.

Procedure: To configure the address completion timeout

Mode: Context CS

Step	Command	Purpose
1	*node*(ctx-cs)[*switch*]# **address-completion timeout** *timeout*	Configures the address completion timeout by specifying the *timeout* in seconds. If not configured, the default address completion timeout is 12 seconds.

Example: Configure address completion timeout

```
node[switch]#address-completion timeout 20
```

Configures the address completion timeout to 20s. A call with an incomplete called party number is dropped 20 seconds after receiving the last called party address update (overlap dialed digit).

Configure default digit collection timeout and terminating character

You can enter called party routing table entries that specify an incomplete number (extended by the T-indicator; refer to section "Called party number routing table" on page 466). When the call router selects such an entry, the call router waits for additional digits and starts the digit collection timeout. When the digit collection timeout elapses or when the user enters the terminating character, the call is placed to the destination specified with the routing table entry.

The digit collection timeout has a default value of 5 seconds, the terminating character is the pound character (#) by default.

> **Note** The digit completion timeout is active when the call router waits for optional digits of a called party number before placing the call to the selected destination. Contrary to this, the address completion timeout, described above, waits for mandatory digits.

Procedure: To configure the digit collection timeout and terminating character

Mode: Context CS

Step	Command	Purpose
1	*node*(ctx-cs)[*switch*]# **digit-collection timeout** *timeout*	Configures the digit collection timeout by specifying the *timeout* in seconds. If not configured, the default digit collection timeout is 5 seconds.
2	*node*(ctx-cs)[*switch*]# **digit-collection terminating-char** *char* *or* **no digit-collection terminating-char**	Configures the digit collection terminating character by specifying the *char*. This can be any character out of **0123456789*#**. The default terminating character is the pound (#). You can also use this command in the no-form to disable that the user can stop digit collection by a terminating character.

Example: Configure address completion timeout

```
node[switch]#digit-collection timeout 3
node[switch]#digit-collection terminating-char *
```

Configures the digit collection timeout to 3s. The digit-collection timeout can be stopped by the user entering the asterisk (*) character.

Configure number prefix for ISDN number types

The called and calling party numbers in an ISDN signaling message are of a defined number type; national, international or unknown. Depending on where the message originates (PSTN, mobile network, PBX) this number type may differ. Table 16 illustrates the three number types.

Table 16. ISDN number types

Type	Format Description	Example
unknown	as dialed with all leading zeros or other prefix numbers	0041 99 888xxxx
national	(area code) (local extension number)	99 888xxxx
international	(country code) (area code) (local extension number)	41 99 888xxxx

The missing prefix in the national and international number types can complicate the call router configuration, so SmartWare offers the possibility to expand these numbers before entering the first call router table.

Note The configured prefix is not removed at the exit of the call router (i.e. when a destination interface is found), but the number has the number type *unknown*.

Procedure: To configure number prefix

Mode: Context CS

Step	Command	Purpose
1	*node*(ctx-cs)[*switch*]# **national-prefix** *prefix*	Adds *prefix* to all E.164 numbers of type national before entering the call router.
2	*node*(ctx-cs)[*switch*]# **international-prefix** *prefix*	Adds *prefix* to all E.164 numbers of type international before entering the call router.

Example: Configure number prefix

```
node[switch]#national-prefix 0041

Input: 99888xxxx Result: 004199888xxxx

node[switch]#international-prefix 00

Input: 4199888xxxx Result: 004199888xxxx
```

Configure call routing tables

Routing tables are identified by names that can be any arbitrary string. For ease of identification the table type is typically used as part of the name.

Call router tables are created by entering the **routing-table** command, which also brings you into the routing table configuration mode. There you can add, modify or delete entries of the routing tables. Refer to the individual table types detailed below on how to configure table entries.

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

Note To remove a specific entry of a table, enter the table configuration mode and use the no-form on a previously entered entry. To remove a whole table, use the **no** form of the **table mode** command.

Create a routing table

A routing table forwards the call to another table, interface or service based on a specific call property like the called party number or the current date. The call router provides a number of different routing table types. A routing table looks like the following:

Type	called-e164	
Name	NATIONAL	
Key	**Destination**	**Function**
001	if USVOIP-A	
001320	if USVOIP-B	
0044	if EUROVOIP	
0049	if EUROVOIP	
default	if DEFACC	ADD-PREFIX

Header

Entries

Figure 63. Routing table outline

Each table contains a header and one or more entries. The header declares the type of the routing table as well as its name.

The name of the routing table is unique inside the context and serves as identifier for referencing the table from other tables or interfaces. The routing table type specifies which call property the table shall examine. Table 17 lists the call properties that can be used as a routing table type.

Table 17. Routing table types

Type	Description
called-e164	Route calls based on the called party E.164 number. Entries of called-e164 tables can use wildcards to summarize routes. Digit collection can be configured on a per-entry basis.

Table 17. Routing table types (Continued)

Type	Description
calling-e164	Route calls based on the calling party E.164 number. Entries of calling-e164 tables can use wildcards to summarize routes.
called-type-of-number	Route calls based on the called party number type. ISDN distinguishes different type of numbers.
calling-type-of-number	Route calls based on the calling party number type. ISDN distinguishes different type of numbers.
called-numbering-plan	Route calls based on the called party numbering plan. ISDN distinguishes different numbering plans.
calling-numbering-plan	Route calls based on the calling party numbering plan. ISDN distinguishes different numbering plans.
called-name	Route calls based on the display name of the called party.
calling-name	Route calls based on the display name of the calling party.
called-ip	Route calls based on the signaling IP address of the destination VoIP peer.
calling-ip	Route calls based on the signaling IP address of the origination VoIP peer.
called-uri	Route calls based on the URI of the destination VoIP peer (for SIP calls: the To-URI).
calling-uri	Route calls based on the URI of the origination VoIP peer (for SIP calls: the From-URI).
calling-pi	Route calls based on the presentation indicator.
calling-si	Route calls based on the screening indicator.
itc	Route calls based on the information transfer capability (bearer capability) to distinguish speech from data calls.
time	Route calls based on the current time of day. .
date	Route calls based on the current date.
day-of-week	Route calls based on the current day of week.

Besides the header (name and type) a routing table contains multiple entries. Each entry specifies a specific value of the routing table type and a destination interface and an optional function. When a call arrives at a routing table, the following procedure is applied:

1. Examine the call property as specified with the routing table type.
2. Select the best matching entry. This means that the key of each of the entries is compared to the call property and the entry that matches best is chosen.
3. Execute the entry. This means executing the referenced function of the entry if specified, and routing the call to the specified destination interface, table or service.

Procedure: To create a routing table and add entries

Mode: Context CS

Step	Command	Purpose
1	node(ctx-cs)[switch]#**routing-table** *table-type table-name*	Create a routing table *table-name* of the specified *table-type*. This enters the table mode where entries can be added or removed. To enter a previously created table from the context CS mode, you may leave away the *table-type*.
2	node(rt-tab)[table-name]#**route** *key* **dest-interface** *if-name function* or node(rt-tab)[table-name]#**route** *key* **dest-table** *table-name function* or node(rt-tab)[table-name]#**route** *key* **dest-service** *service-name function*	Add an entry to the routing table for destination interface *if-name*, destination table *table-name* or destination service *service-name*. Optionally you can specify a *function* (mapping-table or complex function) that shall be executed before the call is routed to the destination interface, table or service. The format of the *key* depends on the type of table. The next sections explain key formats for the different table types.
3		Repeat step 2 to add lines for additional table entries.

Example: Called party number routing table

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table called-e164 NATIONAL
node(rt-tab)[NATIONAL]#route 001 dest-interface USVOIP-A
node(rt-tab)[NATIONAL]#route 001320 dest-interface USVOIP-B
node(rt-tab)[NATIONAL]#route 0044 dest-interface EUROVOIP
node(rt-tab)[NATIONAL]#route 0049 dest-interface EUROVOIP
node(rt-tab)[NATIONAL]#route default dest-interface DEFACC ADD-PREFIX
```

Called party number routing table

The called party number (called-e164) table is used to route calls based on the called party E.164 number in the call set-up message. The call router scans the table to find the longest matching *key* starting with the **first** digit.

Regular Expressions

The *key* of an entry can be either a complete number or a partial number with wildcard digits, represented by a period (.) character. Each (.) represents a wildcard for an individual digit. For example, if the *key* is defined as *888* . . . ., then any called party number beginning with 888, plus at least four additional digits matches this entry.

In addition to the period (.), there are several other symbols that can be used as wildcard characters in the *key*. These symbols provide additional flexibility in designing call routing and decrease the need for multiple entries in configuring number ranges.

The following table shows the wildcard characters that are supported:

Table 18. Wildcard symbols used as keys in E.164 tables (calling-e164, called-e164)

Symbol	Description
.	Indicates a single-digit placeholder. For example, 888 . . . . matches any dialed number beginning with 888, plus at least four additional digits. Note that the key only specifies the prefix. Thus the number may be longer, but also matches.
[ ]	Indicates a range of digits. A consecutive range is indicated with a hyphen (-); e.g. [5-7]. A nonconsecutive range is indicated without a delimiter. For example, [58]. Both can be used in combination; e.g. [5-79], which is the same as [5679]. A (^) symbol may be placed right after the opening bracket to indicate that the specified range is an exclude list. For example, [^01] specifies the same range as [2-9]. **Note:** Only single-digit ranges are supported. You cannot specify the range of numbers between 99 and 102 using [99-102].
( )	Indicates a pattern. For example, 888(2525). It is used in conjunction with the symbol (?), (%) or (+) or when replacing a number in a mapping table.
?	Indicates that the preceding digit or pattern occurred zero or one time. Enter Ctrl-V before entering (?) from your keyboard, since the CLI normally uses the question mark to display help texts.
%	Indicates that the preceding digit or pattern occurred zero or more times. This functions the same as the asterisk (*) used in regular expression. Here the percent (%) symbol is used to be able to handle the asterisk (*) as part of a dialed number.
+	Indicates that the preceding digit or pattern occurred one or more times.
T	Enables digit-collection for this entry. The call router pauses to collect additional dialed digits. The default digit collection timeout is 5 seconds. The collection can be aborted pressing the pound (#) key. **Note:** The terminating character is used only to terminate the timeout and is therefore removed from the dialed number. **Note:** The timeout (T) symbol is only allowed for Called Party Number table entries, while all other wildcards are also allowed for Calling Party Number tables.

The next table shows some examples of how these wildcard symbols are applied to the *key* of a table entry:

Table 19. Wildcard symbols used as keys in E.164 tables (calling-e164, called-e164)

Expression	Description
88825.+	88825, followed by one or more wildcard digits. This expression implies that the number must contain at least 6 digits starting with 88825; for example, 888251, 8882512 or 888251234567890
88825.%	88825, followed by zero or more wildcard digits. This expression implies that the string must contain at least 88825; for example, 88825, 888256, 8882567. **Note:** The ".%" expression postfix can be left away, because the expression is always compared as prefix to the dialed number. Thus each expression automatically contains a ".%" postfix.
88825+	8882, followed by 5 repeated one ore more times; for example, 88825, 888255, or 8882555555555
888(25)+	888, followed by 25 repeated one or more times; for example, 88825, 8882525 or 8882525252525
0?111	An optional 0, followed by 111; for example, 0111, 111, 11123456789
8882[56]...	8882 followed by 5 or 6, plus at least three more wildcard digits.
.%45$	Any number that has a postfix of 45; for example, 45, 045, 0041998882545. **Note:** The dollar sign ($) at the end is used to disable prefix matching.

In addition to wildcard characters, the following characters can also be used in the key of the table entry:

- Asterisk (*) and pound sign (#) – These characters can be used anywhere in the key like other digits, for example, they can be used as the leading character (e.g. *21), which is handled like normally dialed number.
- Dollar sign ($) – Disables prefix matching. Must be used at the end of the dial string.

Digit Collection

Fixed-length dialing plans, in which all the numbers have fixed length, are sufficient for most voice networks, because the telephone number strings are of known lengths. Some voice networks, however, require variable-length dial plans, particularly for international calls, which use telephone numbers of different length. Furthermore some voice networks do not support overlap dialing. In this case the call-router must collect the digits before placing a call to that network with the complete number.

If you enter the timeout T-indicator at the end of the key in a Called-Party Number table, the call router accepts a fixed-length number and then waits for additional dialed digits. The timeout character must be an uppercase T. The following example shows how the T-indicator is set to allow variable-length numbers:

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table called-e164 collect
node(rt-tab)[collect]#route 0041T dest-interface CHVoIP-A
```

In the example above, the call router accepts the digits 0041, and then waits for an unspecified number of additional digits as long as the interdigit timeout has not expired. When the interdigit timeout expires, the router places the call.

The default value for the interdigit timeout is 5 seconds and can be configured using the **digit-collection timeout** command in the context CS configuration mode. You may want to override this default timeout for a specific entry. Just place the timeout in seconds after the T-indicator; e.g. T3 to set the inter digit timeout to 3 seconds for that entry.

The user may press the pound (#) as terminating character to immediately place the call. If the pound (#) character is entered while the router is waiting for additional digits, the pound (#) character is treated as a terminator; it is not treated as part of the dialed number sent across the network. But if the pound (#) character is entered before the router begins waiting for additional digits (meaning that the pound (#) is entered as part of the fixed-length key), then the pound (#) character is treated as a dialed digit.

For example, if the key is configured as 888 . . . . T, then the entire dialed string of 888#2525 is collected, but if the dialed string is 888#252#5, the #5 at the end of the dialed number is not collected, because the final pound (#) character is treated as terminator. You can change the default terminating character using the **digit-collection terminating-char** command in the context CS configuration mode. You may want to override this default terminating character for a specific entry. Just place the character after the timeout and a comma; for example, T3,* to set the terminating character to asterisk (*).

Digit Collection Variants

There are three different ways how a called party routing-table can be used to perform address completion or digit collection. Consider the following examples:

```
routing-table called-e164 TAB-PREFIX
  route 099 dest-interface IF-OUT

routing-table called-e164 TAB-COMPLETE
  route 099.... dest-interface IF-OUT

routing-table called-e164 TAB-COLLECT
  route 099T dest-interface IF-OUT
```

Now assume someone picks up a phone and dials a number using overlap dialing. After picking up the phone, an empty called party number is offered to the routing tables. All three routing tables require the called party number to contain at least the prefix 099. Thus the number is incomplete and the call router waits for additional digits being entered. Now the user presses the digit one (1). The resulting called party number is 0991. The call router is again asked for routing the call to a destination. The TAB-PREFIX table performs a prefix match with its only entry and finds out that the number is long enough, so TAB-PREFIX immediately routes the call to the destination interface IF-OUT. Unlike the first table, TAB-COMPLETE needs at least three more digits. Thus the address is not complete yet and the call router waits for more digits. TAB-COLLECT has enough digits but uses the T-indicator to perform digit-collection. The call router waits for the digit-collection timeout and then places the call to the destination interface IF-OUT.

Note There is a difference between the address completion and the digit collection timeout. The address completion timeout is active when a route is incomplete, e.g. when the dialed number of 0991 is tried to match to the entry *099.….* In this case, the call router cannot forward the call to the destination unless the user enters three more digits. Thus the address completion timeout is active when the call router waits for mandatory digits. The address completion timeout can be configured using the **address-completion timeout** command in the context CS mode. If the user does not enter more

digits, the address completion timeout elapses and the call is dropped. The digit collection timeout is active when a route is complete but a T-indicator is specified on the selected route, e.g. when the dialed number of 0991 is tried to match the entry *099T*. In this case the call router waits for some period of time for the user to enter additional optional digits. The digit collection timeout can be configured using the **digit-collection timeout** command in the context CS mode. If the user does not enter more digits, the digit collection timeout elapses and the call is forwarded to the destination interface.

Example: Simple called party number routing table

The following table routes call based on the called party number. An internal number starting with 5 and containing at least 3 digits is routed to the interface that is connected to the local PBX. An international call to the US is routed to the VoIP interface USVOIP. All other calls (local, national and international) are routed to the interface that is connected to the PSTN.

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table called-e164 DIST
node(rt-tab)[DIST]#route 5.. dest-interface PBX
node(rt-tab)[DIST]#route 001T dest-interface USVOIP
node(rt-tab)[DIST]#route default dest-interface PSTN
```

Example: Digit collection of any number

If you want to route calls from interface A directly to interface B, wanting to collect dialed digits, you have to route calls from interface A to a routing table like the one shown in this example. This table does not require the dialed number to be of any format or length but waits for arbitrary number of digits that can be entered using overlap dialing. This allows the user to enter any number to reach the destination interface.

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table called-e164 COLLECT
node(rt-tab)[COLLECT]#route T dest-interface OUT
```

Example: Called party number routing table explained

Regular expressions are very powerful for E.164 lookups. A routing table always tries to find the table entry with the best matching prefix. Determining the best match is often not a very simple process. There are several rules that define the match quality of a rule for an entered number:

1. A longer rule matches better than a shorter one.
2. An explicitly specified digit matches better than a wildcard.
3. A wildcard that include less possible digits matches better that a wildcard that include more possible digits.

Consider the following routing table:

```
routing-table called-e164 test
  route 1       dest-interface IF1entry #1
  route 1[0-4]  dest-interface IF2entry #2
  route 11      dest-interface IF3entry #3
  route 111T    dest-interface IF4entry #4
```

```
route default dest-interface IF5entry #5
```

Note The numbers that are normally dialed are longer than the prefixes listed in the table test. For example, if the numbering plan is defined using five digits, a user normally dials a number like 12345 to reach a destination. Anyway the lookup result must be the same for en-bloc and for overlap dialing. This example shows how the table is looked up after each overlap-dialed digit and how a destination is finally found. This selected destination is the same as if the user dialed the number en-bloc.

The following table contains a list of overlap-dialed number examples that use the routing table above for a lookup:

Dialed Number	Selected Entry	Description
empty (after picking up the phone without dialing a number beforehand)	—	The number is incomplete for entries #1-#4, which all want at least know whether the number starts with a 1. Thus no entry is selected and the call router waits for additional mandatory digits or drops the call after 12 seconds (address-completion timeout).
1	—	No entry is selected. Though the dialed number matches entry #1 there are other entries that are still incomplete (the entered number is a prefix of the entry). In this state the call router waits for additional mandatory digits or drops the call after 12 seconds (address-completion timeout).
2	#5	No entry matches, so the default entry is selected; the call is placed immediately.
11	—	No entry is selected. Though the dialed number completely matches entry #1, #2 and #3, entry #4 is still incomplete. The call router waits for additional mandatory digits or drops the call after 12 seconds (address-completion timeout).
12	#2	Entry #1 and #2 match the dialed number of 12, but entry #2 matches better because the expression is more precise (longer) than entry #1. Thus the call is immediately routed to interface IF2.
19	#1	Entry #1 is the only that matches. The call is immediately placed.
111	#4	All entries match the dialed number of 111, but entry #4 matches best because the expression is more precise (longer) than entry #1-#3. Entry #4 is selected but the call is not placed immediately because the entry contains the T-indicator. The router waits for additional digits and then places call to interface IF4 when the digit-collection timeout elapses. Note: If the user enters an additional digit during digit-collection on a T-indicator, the router must not change the destination entry anymore.
112	#3	Entry #1, #2 and #3 match the dialed number of 112. Entry #1 has only an expression of one digit while entry #2 and #3 have an expression that specify two digits. Entry #3 matches better than entry #2 because entry #3 explicitly specifies the digits while entry #2 contains a wildcard for the second digit. Thus entry #3 is selected and the call is placed immediately to interface IF3.

Dialed Number	Selected Entry	Description
121	#2	Entry #1 and #2 match the dialed number of 121, but entry #2 matches better. The call is immediately placed to IF2.
191	#1	Only entry #1 matches the dialed number of 191. Thus the call is routed immediately to interface IF1.
1111	#4	The lookup procedure is the same as for dialled number 111. The call router waits for additional digits and places the call after the digit-collection timeout to interface IF4.
1111#	#4	Same as for 1111, but the pound (#) terminates the digit collection; the call is immediately placed to interface IF4.

Calling party number routing table

The calling party number (calling-e164) table is used to route calls based on the calling-e164 in the call setup message. This number in general corresponds to the extension number of a PBX or MSN of an ISDN terminal. The table can be used to route calls from extensions, which have particular call routing requirements (i.e. Terminals which require non VoIP capable ISDN services). The call router looks for the longest match starting with the first digit of the calling party number.

Note The calling party number is sometimes inserted or modified by a PBX. Sometimes there is no calling party number at all. This all depends on the equipment you connect to the device.

Note The T-indicator cannot be used in calling party number tables. (Overlap dialing only makes sense for called party numbers).

Example: Calling party number routing table

This example shows how to create a calling party number routing table that routes calls based on the last three digits of the calling party number (the extension part). The key *.%52[35]$* means that every number that starts with any digit (.) appearing zero or more times (%) followed by 52 and a 3 or 5 matches the entry. For example, the following calling party numbers match the first entry: 0998882523 or 0998882525 or simply 523 or 525.

Note This table does not contain a default entry. All calls where the calling party number does not match to one of the entries are dropped.

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table calling-e164 EXTS
node(rt-tab)[EXTS]#route .%52[35]$ dest-interface breakout
node(rt-tab)[EXTS]#route .%572 dest-interface DefAcc
```

Number type routing table

The calling or called party number type (calling-type-of-number or called-type-of-number) table is used to route calls based on the calling or called party type of number field in the ISDN setup message. This can be used, for example, to differentiate between national and international calls.

Note When you specified a national or international prefix using the commands national-prefix or international-prefix respectively. in the context CS configuration mode, the calling or called party number is extended with the specified prefix and the type-of-number is set to unknown in the incoming interface. Thus an international number can enter the call-router as unknown number.

Note This property is only set by incoming ISDN calls. A call that arrives the call router from a FXS or SIP interface has a number type of Unknown as those interfaces do not support the number type property.

The call router can route calls according to the following number types. These values beside default can be used for the key parameter to create a routing table entry:

- unknown—Unknown number type. This is the default value for calls that arrive through an interface that does not support the number type property.
- international—International number; the number does not include prefix or escape digits.
- national—National number; the number does not include prefix or escape digits.
- network-specific—Network specific number, used to indicate administration or service number specific to the serving network, e.g. used to access an operator.
- subscriber—Subscriber number; the number does not include prefix or escape digits.
- abbreviated—Abbreviated number.

Example: Calling type-of-number routing table

The following example routes calls with an international calling party number to the next table TAB-INCOMING-INT, calls with a national calling party number to the next table TAB-INCOMING-NAT and all other calls to the next table TAB-INCOMING-UNKNOWN.

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table calling-type-of-number TON
node(rt-tab)[TON]#route international dest-table TAB-INCOMING-INT
node(rt-tab)[TON]#route national dest-table TAB-INCOMING-NAT
node(rt-tab)[TON]#route default dest-table TAB-INCOMING-UNKNOWN
```

Numbering plan routing table

The calling party numbering plan or called party numbering plan (calling-number-plan or called-numbering-plan) table is used to route calls based on the calling or called party numbering plan field in the ISDN setup message. This can be used to differentiate calls between numbers of an ISDN, data, telex, national standard or private numbering plan.

Note This call property is only set by incoming ISDN calls. A call that arrives the call router from a FXS or SIP interface has a numbering plan of unknown.

The call router can route calls according to the following numbering plans. These values beside default can be used for the key parameter to create a routing table entry:

- unknown—Unknown numbering plan. This is the default value for calls that arrive through an interface that does not support the numbering plan property.
- isdn-telephony—ISDN/Telephony numbering plan according to CCITT Recommendation E.164/E.163).
- data—Data numbering plan according to CCITT Recommendation X.121.
- telex—Telex numbering plan according to CCITT Recommendation F.69.
- national-standard—Numbering plan according to a national standard.
- private—Any private numbering plan.

Example: Calling numbering-plan routing table

The following example shows how to create a routing table with name NP that routes calls based on the calling part numbering plan. Calls with calling party numbers formed according to the ISDN/Telephony numbering plan are routed to the next table *TAB-INCOMING-ISDN*, calls with calling party numbers formed according to the data numbering plan to the next table *TAB-INCOMING-DATA* and all other calls to the next table *TAB-INCOMING-UNKNOWN*.

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table calling-numbering-plan NP
node(rt-tab)[NP]#route isdn-telephony dest-table TAB-INCOMING-ISDN
node(rt-tab)[NP]#route data dest-table TAB-INCOMING-DATA
node(rt-tab)[NP]#route default dest-table TAB-INCOMING-UNKNOWN
```

Name routing table

The calling display name or called display name (calling-name or called-name) table is used to route calls based on the human-readable name of the calling or called party. The key you specify in a routing table entry must be identical to the display name of the calling or called party for the entry to match.

Note Incoming SIP calls use this call property to store the display name part of the SIP URI. Other interfaces set the display name to the empty string.

Example: Calling display name routing table

This example routes calls based on the display name part of the From-URI of an incoming SIP call.

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table calling-name name
node(rt-tab)[name]#route “John Smith” dest-table TAB-FROM-JOHN
node(rt-tab)[name]#route Administrator dest-table TAB-FROM-ADMIN
node(rt-tab)[name]#route default dest-table TAB-FROM-UNKNOWN
```

IP address routing table

The calling party IP address (calling-ip) table is used to route calls based on the signaling IP address of the originating VoIP peer, e.g. the IP address of the originating H.323 peer terminal. The called party IP address (called-ip) table is used to route calls based on the called IP address property. The called IP address of incoming calls is set to 0.0.0.0 unless modified by a previous mapping table in the routing path. Thus the called IP address table is of limited use only.

You may specify a whole subnet with the key parameter of the routing table entry. The format of the key parameter is ipaddress[/mask-size]; the mask size may be omitted.

Note Incoming SIP and H.323 calls use the calling party IP address property to store the IP address of the remote SIP user agent or H.323 terminal, respectively. Other interfaces like ISDN or FXS set the IP address to 0.0.0.0.

Example: Calling IP address routing table

The following example routes all calls from a remote H.323 terminal in the LAN to the next table *TAB-FROM-LAN* and all other calls to *TAB-FROM-WAN*.

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table calling-ip ip
node(rt-tab)[ip]#route 172.16.32.0/24 dest-table TAB-FROM-LAN
node(rt-tab)[ip]#route default dest-table TAB-FROM-WAN
```

URI routing table

The calling party URI (*calling-uri*) table is used to route calls based on the URI of the originating VoIP peer, e.g. the From-URI of the incoming SIP call. The called party URI (*called-uri*) table is used to route calls based on the *To-URI*. The called URI of incoming calls is not set unless modified by a previous mapping table in the routing path.

You can use regular expressions to specify the parts of an URI that must match in order to route the call to a specified destination.

The following example shows how to create a routing table to route all SIP calls from John Smith to the next table *TAB-FROM-JOHN* while all other calls are routed to the next table *TAB-FROM-UNKNOWN*.

Mode: Context CS

Step	Command	Purpose
1	**node(ctx-cs)[switch]#routing-table calling-uri** name	Creates a routing-table that examines the From-URI of an incoming SIP call
2	**node(rt-tab)[***name***]#route sip:john\.smith@.% dest-table TAB-FROM-JOHN**	Routes all SIP calls from john.smith@<anywhere> to the table TAB-FROM-JOHN. Note that the dot (.) between john and smith must be escaped with a backslash (\), because the dot (.) means 'any character' in a regular expression.
3	**node(rt-tab)[***name***]#route default dest-table TAB-FROM-UNKNOWN**	Routes all other SIP calls to the table TAB-FROM-UNKNOWN

Presentation Indicator Routing Table

The presentation indicator (calling-pi) table is used to route calls based on the presentation indicator of the calling party number. A user that doesn't want its number being displays sets the presentation indicator to restricted. There is no presentation indicator on the called party number. Thus you cannot create a called-pi table.

Note Incoming ISDN calls set the presentation indicator according to the received ISDN Setup message. Incoming H.323 calls only set the presentation indicator transparently when Octet3a handling is enabled. Other interfaces set the presentation indicator to allowed.

The call router can route calls according to the following presentation indicators. These values beside default can be used for the key parameter to create a routing table entry:

- allowed—Presentation of the calling party number is allowed. This is the default value for calls that arrive through an interface that does not support presentation indicators.
- restricted—Presentation of the calling party number is restricted.
- interworking—The calling party number is not available due to interworking.

Note At the originating user-network interface, the presentation indicator is used for indicating the intention of the calling user for the presentation of the calling party number to the called user. You possibly want to remove the calling party number when the user set the presentation indicator to restricted. To achieve this, route restricted calls to a mapping table that sets the calling-e164 to the empty string ("") as it is shown in the example below.

Example: Presentation indicator routing table

This example uses a pseudo routing table that just forwards all calls to the interface *IF-OUT* but first executes the mapping table *NO-CNPN*. This mapping table examines the presentation indicator and modifies the calling party number. If the presentation indicator is restricted, the calling party number is cleared. For all other presentation indicator values, the calling party number is not modified.

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table calling-pi PI
node(rt-tab)[PI]#route default dest-interface IF-OUT NO-CNPN
node(rt-tab)[PI]#exit
node(rt-tab)[switch]#mapping-table calling-pi to calling-e164 NO-CNPN
node(map-tab)[NO-CNPN]#map restricted to ""
```

Screening Indicator Routing Table

The screening indicator (*calling-si*) table is used to route calls based on the screening indicator of the calling party number. A network validates the calling party's number and puts the validation result to the screening indicator. This allows a network to transparently pass the calling party number set by the calling user, but tell the called user whether or not the number selected by the calling party really belongs to him or her.

Note Incoming ISDN calls set the screening indicator according to the received ISDN Setup message. Incoming H.323 calls only set the screening indicator transparently when Octet3a handling is enabled. Other interfaces set the screening indicator to user-not-screened.

The call router can route calls according to the following screening indicators. These values beside default can be used for the key parameter to create a routing table entry:

- user-not-screened—The calling party number is provided by the user but not screened by the network. Thus the calling party possibly send a number that is not owned by the calling party. (This is the default value for calls that arrive through an interface that does not support presentation indicators).
- user-passed—The calling party number is provided by the user, screened by the network and really belongs to the calling party.
- user-failed—The calling party number is provided by the user, screened by the network and does not belong to the calling party.
- network—The calling party number, because it is provided by the network, can be trusted.

Note You possibly want to remove the calling party number when the calling party number is not screened or screening failed. To achieve this, route these calls to a mapping table that sets the calling-e164 to the empty string (“”). If you want to drop calls when the calling party number is not screened or screening failed, use the routing destination none.

Example: Screening indicator routing table

A call that is routed to this table is examined for the screening indicator of the calling party number. If the calling party provided a number that was not accepted by the network (user-failed), we drop the call. Else we route the call to the interface *IF-OUT* but first execute the mapping table *NO-CNPN*. This mapping table again examines the screening indicator. If the user provided a number that was not screened by the network, the table sets the calling party number to an empty string. For all other screening indicator values, the calling party is not modified.

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table calling-si SI
node(rt-tab)[SI]#route user-failed none
node(rt-tab)[SI]#route default dest-interface IF-OUT NO-CNPN
node(rt-tab)[SPI]#exit
node(rt-tab)[switch]#mapping-table calling-si to calling-e164 NO-CNPN
node(map-tab)[NO-CNPN]#map user-not-screened to “”
```

Information transfer capability routing table

The information transfer capability (*itc*) table is used to route calls based on the information transfer capability field of the bearer capability information element in the ISDN setup message. This can be used to differentiate between ISDN data services and ISDN speech connections.

Note Terminals connected to analog extensions (e.g. of a PBX) do not supply information transfer capability values in their call set-up, so it is up to the configuration of the analog port on the Terminal Adapter, NT or PBX to insert this value. The configuration of this value is however often omitted or wrong. The ITC value may therefore not be a reliable indication to differentiate between analogue speech, audio or Fax Group 3 connections. Also, calls from SIP and FXS interfaces do not differentiate between bearer capabilities. They always set the information transfer capability property to *3.1kHz Audio*.

The call router can route calls according to the following information transfer capabilities. These values beside default can be used for the key parameter to create a routing table entry:

- speech—Voice terminals (Telephones)
- unrestricted-digital—Unrestricted digital information (64kBit/s)
- restricted-digital—Restricted digital information (64kBit/s)
- 3k1-audio—Transparent 3.1kHz audio channel. This is the default value set by interfaces that do not support the ITC property.
- 7k-audio—Transparent 7.1kHz audio channel
- video—Video conference terminals

Example: Information transfer capability routing table

The following example creates an itc routing table that routes all unrestricted digital calls to the interface *IF-ACCESS*, all 7kHz audio calls to the interface *IF-LOCAL-BREAKOUT*, all video calls to the interface *IF-ACCESS* and all other calls to the interface *IF-VOIP-CARRIER-A*.

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table itc ITC
node(rt-tab)[ITC]#route unrestricted-digital dest-interface IF-ACCESS
node(rt-tab)[ITC]#route 7k-audio dest-interface IF-LOCAL-BREAKOUT
node(rt-tab)[ITC]#route video dest-interface IF-ACCESS
node(rt-tab)[ITC]#route default dest-interface IF-VOIP-CARRIER-A
```

Call-router support for redirecting number and redirect reason

The call.router can be used to manipulate and make routing decisions depending on the call-control redirecting number and redirect reason properties. The following command creates a call-router routing-table, which uses the redirecting number for routing decisions. The contents of the routing table are the same as for any other E.164 number based call-router routing table.

Mode: context cs

Step	Command	Purpose
1	[*name*] **(ctx-cs)**[router]**# routing-table called-redir-e164 <table-name>**	Creates a redirecting number routing table.

The following command creates a redirect-reason call-router routing-table. Possible reason values for routing decisions are:

- cd: Call deflection
- cfb: Call forwarding on busy
- cfd: Call forwarding by called DTE
- cfnr: Call forwarding on no reply
- cfu: Call forwarding unconditional
- ooo: Called DTE out of order

- default: Any other unhandled case

Mode: context cs

Step	Command	Purpose
1	[*name*] **(ctx-cs)**[*router*]**# routing-table calling-redir-reason <table-name>**	Creates a redirect reason routing table.

Both the redirecting-number and the redirect-reason can also be used in any call-router mapping tables.

Time of day routing table

The time table is used to route calls based upon the current system time during one day, i.e. an 24hr. period from midnight to midnight. Times are matched within the ranges defined in the time routing table.

The key parameter of the routing table entry has the format: *hh:mm:ss-hh:mm:ss*

The full range must be specified. The range must not cross a day boundary at midnight.

Example: Time of day routing table

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table time WORKDAY
node(rt-tab)[WORKDAY]#route 08:00:00-16:59:59 dest-table TAB-BEST-QUAL
node(rt-tab)[WORKDAY]#route 17:00:00-20:59:59 dest-interface IF-VOIP-A
node(rt-tab)[WORKDAY]#route 21:00:00-23:59:59 dest-interface IF-VOIP-B
node(rt-tab)[WORKDAY]#route 00:00:00-07:59:59 dest-interface IF-VOIP-B
```

Day of Week Routing Table

The day-of-week table is used to route calls according to the day of the week. The days are defined by the long lowercase names monday; tuesday; wednesday; thursday; friday; saturday; and sunday. To configure weekday routing table entries use the following commands starting in the CS context configuration mode.

Example: Day of week routing table

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table day-of-week TAB-DAY
node(rt-tab)[TAB-DAY]#route saturday dest-table TAB-LEAST-COST
node(rt-tab)[TAB-DAY]#route sunday dest-table TAB-LEAST-COST
node(rt-tab)[TAB-DAY]#route default dest-interface IF-VOIP
```

Date routing table

The date table is used to route calls according to the current system date. It can be used, for example, to represent holidays in the routing decision tree. The table matches exact dates or date ranges.

The key parameter of the routing table entry has the format: *dd:mm:yyyy-dd:mm:yyyy*

The full range must be specified.

Example: Date routing table

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table day-of-week HOLIDAY2001
node(rt-tab)[HOLIDAY~]#route 01.01.2001-02.01.2001 dest-table TAB-HOL
node(rt-tab)[HOLIDAY~]#route 05.01.2001-05.01.2001 dest-table TAB-HOL
node(rt-tab)[HOLIDAY~]#route 24.12.2001-31.12.2001 dest-table TAB-HOL
```

```
node(rt-tab)[HOLIDAY~]#route default dest-interface IF-VOIP
```

Deleting routing tables

To remove individual routing tables you can use the **no** form of the **routing table** command. Alternatively you can remove specific entries of a routing table by entering the routing table configuration mode and use the **no** form of the **route** command.

Procedure: To delete an entry from a routing table

Mode: Context CS

Step	Command	Purpose
1	*node*(ctx-cs)[*switch*]#**routing-table** *table-name*	Enter the routing table from which you want to remove an entry. **Note:** You do not have to enter the type of the table when just entering it. The type must only be specified when creating a table.
2	*node*(rt-tab)[*table-name*]#**no route** *key*	Remove the entry with the specified *key*.
3		Repeat Step 2 to remove additional entries.

Example: Remove entries from a routing table

The running-config shows the following table:

```
routing-table called-e164 MY-TABLE
  route 10 dest-interface IF1
  route 11 dest-interface IF2
  route 12 dest-interface IF3
  route default dest-interface IF4
```

To remove the first two entries from the table enter the following commands:

```
node(cfg)#context cs
node(ctx-cs)[switch]#routing-table MY-TABLE
node(rt-tab)[MY-TABLE]#no route 10
node(rt-tab)[MY-TABLE]#no route 11
```

The resulting running-config is:

```
routing-table called-e164 MY-TABLE
  route 12 dest-interface IF3
  route default dest-interface IF4
```

Procedure: To delete an entire routing table

Mode: Context CS

Step	Command	Purpose
1	*node*(ctx-cs)[*switch*]#**no routing-table** *table-name*	Delete the routing table *table-name*. **Note:** You do not have to enter the type of the table when just deleting it. The type must only be specified when creating a table.

Example: Remove an entire routing table

```
node(cfg)#context cs
node(ctx-cs)[switch]#no routing-table MY-TABLE
```

Configure mapping tables

Mapping tables are used to modify the call setup message and thus influence the routing decision and or the outgoing setup message leaving the call router. Mapping tables are identified by a name (string) and referenced in the routing tables for execution. Like the routing table, a mapping table finds the best matching entry and executes it; but unlike the routing table, the execution means manipulating a call property. Thus a mapping table always examines one call property (the input-type) and changes another property (the output-type). As for the routing tables the call router provides a number of different mapping table types. A mapping table looks like the following:

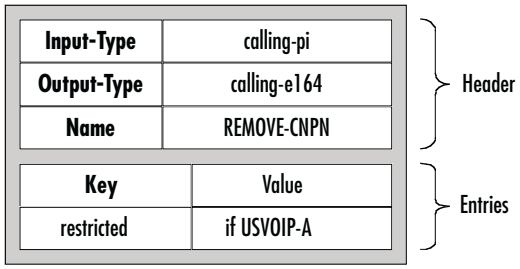


Figure 64. Mapping table outline

Each table contains a header and one or more entries. The header declares the input and output-type of the mapping table as well as its name.

Unlike a routing table, a call property pair characterizes a mapping table, the input and output-type. While the input-type defines which call property is examined by the call router, the output-type defines which property is modified once the best matching entry is found, for example, you may want to find a best matching entry in a mapping table based on the presentation indicator and, once found, you want to manipulate the calling party number of the call. In this case you chose an input-type of calling-pi and an output-type of calling-e164

There are three different kinds of call properties, calling party properties, called party properties and generic properties. When a call setup is to be routed by the call router, most properties appear as calling and as called party properties, for example, you have a calling party number and a called party number to examine. There are other properties (e.g. the information transfer capability), which is a property of the whole call and not of either party.

You can create a mapping table that examines and modifies a specific kind of property, e.g. the called party number. In this case you have to specify an input-type of called-e164 and an output-type of called-164. If you want to replace both, the called and the calling party property with the same mapping table, you can create a mapping table with input-type e164 and output-type e164, i.e. without prefixing the input- and output-type with "called-".

The name of the mapping table is unique inside the context and serves as identifier for referencing the mapping table from other routing tables. Almost every type explained with the routing table above can be used as input and output-type of a mapping table.

Table 20. Mapping table types

Type	Description Input-Type	Description Output-Type
called-e164	Select an entry based on the called party E.164 number. You can use wildcards to summarize entries.	Modifies the called party E.164 number. If the input-type is a called-e164 or a calling-e164 type, you can use replacement operators to use parts of the lookup key.
calling-e164	Select an entry based on the calling party E.164 number. You can use wildcards to summarize entries.	Modifies the calling party E.164 number. If the input-type is a called-e164 or a calling-e164 type, you can use replacement operators to use parts of the lookup key.
e164	If the output-type is also a generic kind of property, this mapping table is applied to both, this calling-e164 and the called-e164 property.	If the input-type is also a generic kind of property, this mapping table is applied to both, the calling-e164 and the called-e164 property.
called-type-of-number	Select an entry based on the called party number type. ISDN distinguishes different type of numbers.	Sets the called party number type.
calling-type-of-number	Selects an entry based on the calling party number type. ISDN distinguishes different type of numbers.	Sets the calling party number type.
type-of-number	If the output-type is also a generic kind of property, this mapping table is applied to both, this calling-type-of-number and the called- type-of-number.	If the input-type is also a generic kind of property, this mapping table is applied to both, the calling-type-of-number and the called-type-of-number property.
called-numbering-plan	Selects an entry based on the called party numbering plan. ISDN distinguishes different numbering plans.	Sets the called party numbering plan.
calling-numbering-plan	Selects an entry based on the calling party numbering plan. ISDN distinguishes different numbering plans.	Sets the calling party numbering plan.
numbering-plan	If the output-type is also a generic kind of property, this mapping table is applied to both, this calling-numbering-plan and the called-numbering-plan.	If the input-type is also a generic kind of property, this mapping table is applied to both, the calling-numbering-plan and the called-numbering-plan property.

Table 20. Mapping table types (Continued)

Type	Description Input-Type	Description Output-Type
called-name	Selects an entry based on the display name of the called party.	Sets the display name of the called party.
calling-name	Selects an entry based on the display name of the calling party.	Sets the display name of the calling party.
name	If the output-type is also a generic kind of property, this mapping table is applied to both, this calling-name and the called-name.	If the input-type is also a generic kind of property, this mapping table is applied to both, the calling-name and the called-name property.
called-ip	Selects an entry based on the remote signaling IP address of the destination VoIP peer.	Sets the remote IP address of the destination VoIP peer.
calling-ip	Selects an entry based on the remote signaling IP address of the origination VoIP peer.	Sets the remote IP address of the origination VoIP peer.
ip	If the output-type is also a generic kind of property, this mapping table is applied to both, this calling-ip and the called-ip.	If the input-type is also a generic kind of property, this mapping table is applied to both, the calling-ip and the called-ip property.
calling-pi	Selects an entry based on the presentation indicator.	Sets the presentation indicator.
calling-si	Selects an entry based on the screening indicator.	Sets the screening indicator.
itc	Selects an entry based on the information transfer capability (bearer capability) to distinguish speech from data calls.	Sets the information transfer capability.
time	Route calls based on the current time of day.	Cannot be set.
date	Route calls based on the current date.	Cannot be set.
day-of-week	Route calls based on the current day of week.	Cannot be set.

Besides the header (name, input and output-type) a mapping table contains multiple entries. Each entry specifies a specific value of the routing table input-type and a value that shall be applied to the call property specified with the output-type of the table. When a call arrives a mapping table, the following procedure is applied:

1. Examine the call property as specified with the mapping table input-type.
2. Select the best matching entry. This means that the key of each of the entries is compared to the call property and the entry that matches best is chosen.
3. Execute the entry. This means replacing the property specified with the output-type of the mapping table with the value of the selected entry.

Let's examine the mechanism of mapping tables in more detail. Figure 65 shows three mapping tables and a call that is routed through this mapping table. The call contains various call properties that are examined and modified by the mapping tables:

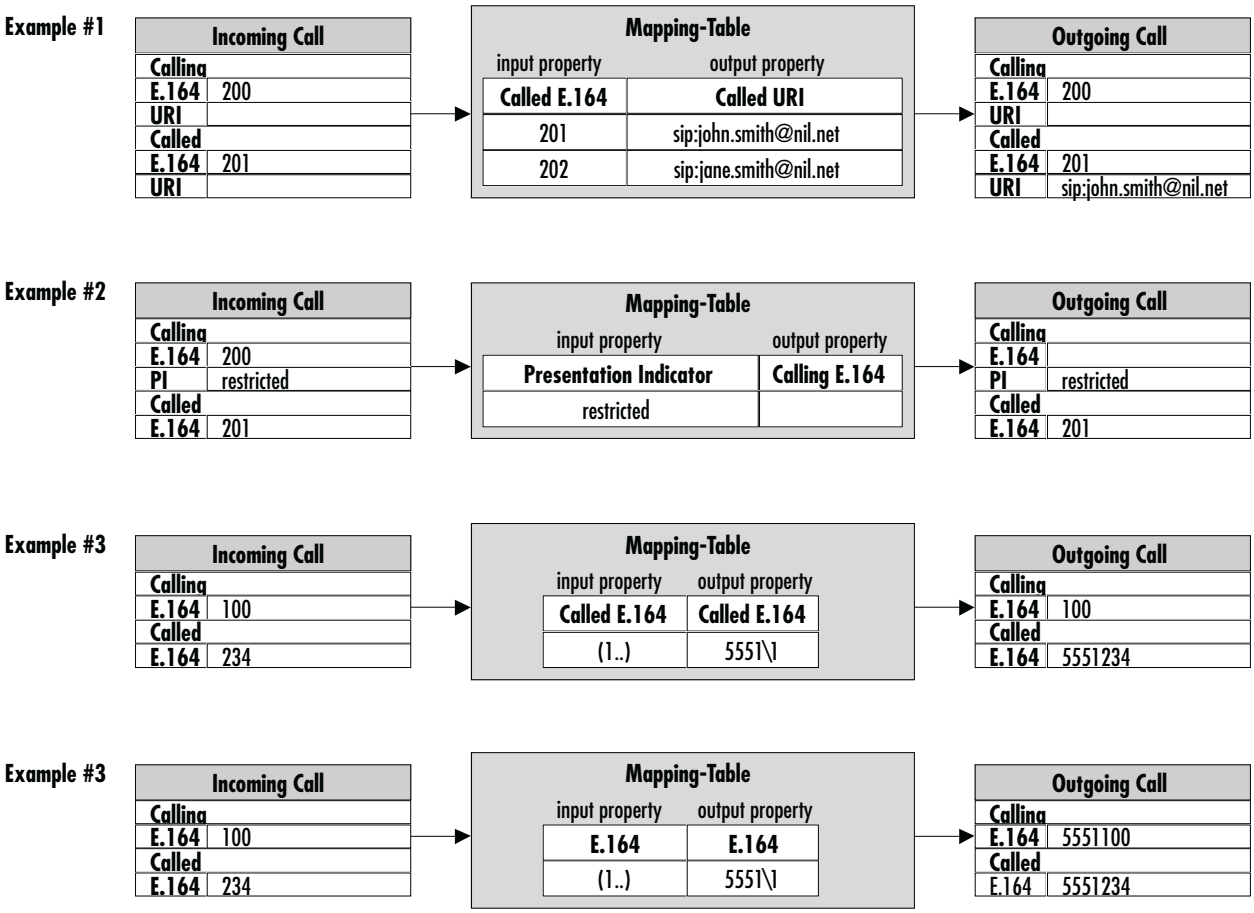


Figure 65. Mapping table examples

Example #1: This example shows a mapping table that selects the best matching entry based on the called party number of a call and, once found, sets the called party URI. In the example a call arrives to the mapping table with a called party number of 201. The mapping table selects the first entry, which matches best, and sets the called party URI of the call to *sip:john.smith@nil.net*.

Example #2: This example shows a mapping table that selects the best matching entry based on the presentation indicator and, once found, sets the called party number. In the example a call arrives to the mapping table with a presentation indicator of *restricted*. The mapping table selects the only entry (which matches) and sets the called party number to the empty string.

Example #3: This example shows a mapping table that selects the best matching entry based on the called party number and, once found, changes the same property, the called party number. This is a very powerful method to manipulate numbers using regular expressions. In this example a call arrives to the mapping table with a called party number of 234. The mapping table selects the only entry (which matches) and adds a prefix of 5551 to the called party number.

Example #4: This example shows the same input call properties as in example #3. The mapping table is also almost the same, but unlike in the previous example, here we don't look for a specific number type (e.g. called party number, calling party number), but for any E.164 number property of the call. The output property is also a generic number. In this case the mapping table replaces both, the calling and the called party number. For example, the mapping table applies its algorithm to all E.164 numbers of the call.

> **Note** Like a routing table, a mapping table selects the best matching entry based on the input property type. This is done using the best matching prefix method for E.164 numbers and string compare for other properties. Unless you don't have a default entry in a mapping table, no action is performed when no match can be found and the call is not dropped. In the above example #3, if a call arrives the mapping table with a called party number of 200, which does not match the entry (1..), the called party number is not changed.

Procedure: To create a mapping table and add entries

Mode: Context CS

Step	Command	Purpose
1	node(ctx-cs)[switch]#**mapping-table** *input-type* **to** *output-type table-name*	Create a mapping table *table-name* that examines the call property specified with *input-type* and modifies the call property specified with *output-type*. This enters the table mode where entries can be added or removed. To enter a previously created table from the context CS mode, you may leave away the *input-type* and *output-type*.
2	node(map-tab)[table-name]#**map** *key* **to** *value*	Add an entry to the mapping that sets the *output-type* call property to *value* if the *output-type* call property matches the *key*. The format of the *key* depends on the *input*-type of the table, and the format of the *value* depends on the *output-type* of the table.
3		Repeat step 2 to add lines for additional table entries.

Example: Called and calling party manipulation mapping table

```
node(cfg)#context cs
node(ctx-cs)[switch]#mapping-table e164 to uri SETURI
node(rt-tab)[SETURI]#map 100 to sip:john.smith@nil.net
node(rt-tab)[SETURI]#map 101 to sip:jane.smith@nil.net
```

E.164 to E.164 Mapping Tables

As with routing tables you can use regular expressions when selecting an entry in a mapping table based on a calling or called party number. If the output property type of a mapping table is also a calling or called party number, you may use parts of the matched expressions when building the modified number as shown in example #3 above.

Detailed Example: You have an internal dial plan that uses three digit numbers starting with a 2 (e.g. 200, 201, etc.). So when an internal subscriber makes a call, its calling party number contains three digits.

1. You want to route calls to the public switched telephone network (PSTN), that is reachable over and ISDN interface. From the PSTN provider you have an assigned number range from 099-8882500 to 099-8882599.
2. You want to pass the last two digits of your internal subscribers when they are making calls to the PSTN. Thus subscriber 244 should make a call to the PSTN using a calling party number of 099-8882544.

To achieve this, create a mapping table that looks like the following:

```
node(cfg)#context cs
node(ctx-cs)[switch]#mapping-table calling-e164 to calling-e164 MAP-PSTN
node(rt-tab)[MAP-PSTN]#map 2(..) to 09988825\1
```

When a call reaches this table with a calling party number of 244, this number is tried to match to the entries of this table:

```
2(44) matches 2(. .)
```

Thus the only entry is selected and executed. This means setting the calling party number to 09988825\1. The last part of the value (a backslash followed by a single digit number) is a placeholder and means that the first pattern (expression in brackets) of the key shall be used instead.

Thus the called party number is replaced with the specified prefix 09988825 concatenated with the bracketed pattern in the key (44). The result is 0998882544.

Like this you can use brackets around any party of the expression of the key and use the part that matches to this bracket in the value you set.

Example: Mapping table to add a prefix to the called party number

Input:called-e164 = 0998882525

Output:called-e164 = *50998882525

```
node(cfg)#context cs
node(ctx-cs)[switch]#mapping-table called-e164 to called-e164 ADD-PFX
node(rt-tab)[ADD-PFX]#map (.%) to *5\1
```

The input 0998882525 matches the expression (.%) – any character repeated zero or more times.

The first bracket encloses the whole number: (.%) == (0998882525) -> \1 = 0998882525

The output is built as concatenation of *5 and the first bracket \1.

The called party number is set to *50998882525

Example: Mapping table to remove a prefix from the called party number

Input:called-e164 = *50998882525

Output:called-e164 = 0998882525

```
node(cfg)#context cs
node(ctx-cs)[switch]#mapping-table called-e164 to called-e164 REM-PFX
node(rt-tab)[REM-PFX]#map *5(.%) to \1
```

The input *50998882525 matches the expression *5(.%) – the prefix *5 followed by any character repeated zero or more times.

The first bracket encloses the number after the prefix: *5(.%) == *5(0998882525) -> \1 = 0998882525

The output is built from the first bracket \1.

The called party number is set to 0998882525.

Example: Mapping table to truncate the called party number

Input:called-e164 = 0998882525

Output:called-e164 = 525

```
node(cfg)#context cs
node(ctx-cs)[switch]#mapping-table called-e164 to called-e164 TRUNC
node(rt-tab)[TRUNC]#map .%(...) to \1
```

The input 0998882525 matches the expression .%(…) – any character repeated zero or more times followed by three mandatory digits.

The first bracket encloses the last three digits: .%(…) == 0998882(525) -> \1 = 525

The output is built from the first bracket \1.

The called party number is set to 525.

Example: Mapping table to remove the calling party number when restricted

Input:calling-e164 = 0998882525; calling-pi = restricted

Output:calling-e164 = “”; calling-pi = restricted

```
node(cfg)#context cs
node(ctx-cs)[switch]#mapping-table calling-pi to calling-e164 REM-CNPN
node(rt-tab)[REM-CNPN]#map restricted to “”
```

The input (presentation indicator) restricted matches the expression restricted.

The output (calling party number) is an empty string (“”).

The calling party number is cleared.

Example: Mapping table to replace the calling party number with the called party number

If you route a call to an FXS interface, the Bellcore standard only allows to signal the calling party number to the connected analog terminal instead of the called party number. Thus you cannot communicate e.g. which extension is being called at a destination PBX. This command allows sending the called party number as calling party number property. The called party number still remains the same.

Input:calling-e164 = 0998882525; called-e164 = 0778881111

Output:calling-e164 = 0778881111; called-e164 = 0778881111

```
node(cfg)#context cs
node(ctx-cs)[switch]#mapping-table called-e164 to calling-e164 COPY-PN
node(rt-tab)[COPY-PN]#map (.%) to \1
```

The input called party number 0778881111 matches the expression (.%) – any character repeated zero or more times.

The first bracket encloses the last whole called party number: (.%) == (0778881111) -> \1 = 0778881111

The output (calling party number) is built from the first bracket \1.

The calling party number is set to 0778881111.

Custom SIP URIs from called-/calling-e164 properties

The regular expression engine used for mapping-tables and routing-tables in the Call Router can handle not only handle digits, but whole strings. You can construct custom SIP URIs from call leg properties as called and calling e.164. See the following examples.

Example 1: Use regular expressions to create mapping tables that map the called-party-e164 number to the called-party-URI for SIP calls. The following example shows how to build a SIP To-URI from the called-party number.

Mode: Context CS

Step	Command	Purpose
1	**node(ctx-cs)[switch]#mapping-table called-e164 to called-uri** name	Creates a mapping table that examines the called-party number and sets the called-party URI (To-URI).
2	**node(map-tab)[***name***]#map (.+) to sip:user_\1@example.com**	If the called-party number exists (at least one digit) the called-party URI is set to user_<called-e164>@example.com
3	**node(map-tab)[***name***]#map default to sip:anonymous@example.com**	If the called-party number does not exist (calls that don't match to the rule of step 2), the called-party URI is set to anonymous@example.com

Example 2: Use a mapping table to set the display name field of To-URIs for outgoing SIP calls from the called-party number of an incoming call. The following example shows how to set the called-party name based on the called-party number.

Mode: Context CS

Step	Command	Purpose
1	**node(ctx-cs)[switch]#mapping-table called-e164 to called-name** name	Creates a mapping table that examines the called-party number and sets the called-party name.
2	**node(map-tab)[***name***]#map (.%) to \1**	Copies the whole called-party number to the called-party name.

Other mapping tables

Example: Mapping table to set the called party number type to international (unconditionally)

Input:called-e164 = 0041998882525; calling-type-of-number = unknown

Result:called-e164 = 0041998882525; calling-type-of-number = international

```
node(cfg)#context cs
node(ctx-cs)[switch]#mapping-table called-type-of-number to called-type-of-number
SET-INT
node(rt-tab)[SET-INT]#map default to international
```

Any called party number type matches the default entry. Note that the input-type of the table does not matter when the mapping table contains only the default entry. Anyway an input-type must be specified when creating the mapping table.

The output (called party number type) is international.

The called party number type is set to international.

Example: Mapping table to replace called party numbers (translation table)

```
node(cfg)#context cs
node(ctx-cs)[switch]#mapping-table called-e164 to called-e164 TRANS
node(rt-tab)[TRANS]#map 550 to 250
node(rt-tab)[TRANS]#map 551 to 251
node(rt-tab)[TRANS]#map 552 to 252
node(rt-tab)[TRANS]#map 553 to 253
node(rt-tab)[TRANS]#map 554 to 254
node(rt-tab)[TRANS]#map 555 to 255
node(rt-tab)[TRANS]#map 556 to 256
node(rt-tab)[TRANS]#map 557 to 257
node(rt-tab)[TRANS]#map 558 to 258
node(rt-tab)[TRANS]#map 559 to 259
```

Note The translation table above can be reduced using regular expressions.

```
node(cfg)#context cs
node(ctx-cs)[switch]#mapping-table called-e164 to called-e164 TRANS
node(rt-tab)[TRANS]#map 55(.) to 25\1
```

Deleting mapping tables

To remove individual mapping tables you can use the **no** form of the **mapping table** command. Alternatively you can remove specific entries of a mapping table by entering the mapping table configuration mode and use the **no** form of the **map** command.

Procedure: To delete an entry from a mapping table

Mode: Context CS

Step	Command	Purpose
1	*node*(ctx-cs)[*switch*]#**mapping-table** *table-name*	Enter the mapping table from which you want to remove an entry. **Note:** You do not have to enter the type of the table when just entering it. The type must only be specified when creating a table.
2	*node*(map-tab)[*table-name*]#**no map** *key*	Remove the entry with the specified *key*.
3		Repeat Step 2 to remove additional entries.

Example: Remove entries from a mapping table

The running-config shows the following table:

```
mapping-table called-e164 to called-e164 MY-TABLE
  map 10 to 20
  map 11 to 21
  map 12 to 22
  map 13 to 23
```

To remove the first two entries from the table enter the following commands:

```
node(cfg)#context cs
node(ctx-cs)[switch]#mapping-table MY-TABLE
node(map-tab)[MY-TABLE]#no map 10
node(map-tab)[MY-TABLE]#no map 11
```

The resulting running-config is:

```
mapping-table called-e164 to called-e164 MY-TABLE
  map 12 to 22
  map 13 to 23
```

Procedure: To delete an entire mapping table

Mode: Context CS

Step	Command	Purpose
1	*node*(ctx-cs)[*switch*]#**no mapping-table** *table-name*	Delete the mapping table *table-name*. **Note:** You do not have to enter the type of the table when just deleting it. The type must only be specified when creating a table.

Example: Remove an entire mapping table

```
node(cfg)#context cs
node(ctx-cs)[switch]#no mapping-table MY-TABLE
```

Creating complex functions

Complex functions allow combining mapping tables, which need to be executed in sequence. This is useful if, for example, the calling and the called party number have to be modified in the same step. Complex function names can be any arbitrary string.

Procedure: To create a complex number manipulation function

Mode: Context CS

Step	Command	Purpose
1	*node*(ctx-cs)[*switch*]#**complex-function** *function-name*	Create a complex function *function-name*.
2	node(func)[function-name]#**execute** *function* or node(func)[function-name]#**execute** *index function*	Add or inserts an entry to the complex function. *function* can be another complex function or a mapping table that shall be executed. Note: Unlike routing and mapping tables, complex functions are ordered lists of entries, which means that the entries are executed in the order of appearance. You can optionally specify an *index* of where to insert the function.
3		Repeat step 2 to add lines for additional functions to execute.

Example: Create a complex function

```
node(cfg)#context cs
node(ctx-cs)[switch]#complex function CARRIER-TO-LOCAL
node(func)[CARRIER~]#execute 1 TRUNCATE-3-CNPN
node(func)[CARRIER~]#execute 2 DDI-TO-PISN
```

Deleting complex functions

To remove individual complex functions you can use the **no** form of the **complex function** command. Alternatively you can remove specific entries of a complex function by entering the complex function configuration mode and use the **no** form of the **execute** command.

Procedure: To delete an entry from a complex function

Mode: Context CS

Step	Command	Purpose
1	*node*(ctx-cs)[*switch*]#**complex-function** *function-name*	Enter the complex function from which you want to remove an entry.
2	*node*(func)[*table-name*]#**no execute** *index*	Remove the entry with the specified index.
3		Repeat Step 2 to remove additional entries.

Example: Remove entries from a complex function

The running-config shows the following complex function:

```
complex function MY-FUNC
  execute 1 MAP1
  execute 2 MAP2
  execute 3 MAP3
  execute 4 MAP4
```

To remove the first two entries from the complex function enter the following commands. Pay attention on the index. When removing the first entry, the MAP2 function becomes entry with index 1. Thus you have to specify index 1 twice to remove the first two entries:

```
node(cfg)#context cs
node(ctx-cs)[switch]#complext-function MY-FUNC
node(func)[MY-FUNC]#no execute 1
node(func)[MY-FUNC]#no execute 1
```

The resulting running-config is:

```
complex function MY-FUNC
  execute 1 MAP3
  execute 2 MAP4
```

Procedure: To delete an entire complex function

Mode: Context CS

Step	Command	Purpose
1	*node*(ctx-cs)[*switch*]#**no complex-function** *function-name*	Delete the complex function *function-name*.

Example: Remove an entire complex function

```
node(cfg)#context cs
node(ctx-cs)[switch]#no complex-function MY-FUNC
```

Digit collection & sending-complete behavior

Sending-Complete

The call-router can be configured how to handle address-complete indications (e.g. ISDN Sending-Complete IE). On ISDN, H.323 and SIP interfaces, an **address-complete-indication** command controls the tunneling of **address-complete** indications separately for incoming and outgoing calls.

Each interface can be configured to pass transparently address-complete indications, or to explicitly insert or remove it for incoming and outgoing calls. In addition the behavior of the call-router can be configured.

Some call signaling protocols allow a user to dial a destination by using the overlap-sending procedure. These protocols include analog telephony (FXS/FXO), ISDN and H.323. ISDN and H.323 support address-complete indication using the Sending-Complete Information Element. Other protocols wait on a timeout or send a terminating character, e.g. pound (#), to indicate address-completion. The following commands control the tunneling of address-complete indications from the signaling protocol to the internal call-control representation and vice-versa. In addition, the context cs introduces command extensions that control the address-completion handling in the internal call-router.

Ingress interface

On the ingress interface (ISDN, H.323 or SIP) the **address-complete-indication accept <type>** command configures how to map the address-complete indication of the signaling-protocol to the internal call-control. There the indication can be used for call-routing (see below) or mapped again to an address-complete indication on an egress interface (see below).

The possible values for the type argument are:

- **transparent**: Transparently passes an address-complete indication (e.g. ISDN Sending Complete Information Element) to the call-control.
- **set**: Always sets the address-complete indication flag towards the call-control even when no such indication is received from the calling party. This configuration can be used to disable overlap-sending on an interface.
- **clear**: Never signals the address-complete indication towards the call-control even when the indication is received from the calling party. This configuration may be used in rare cases to solve interoperability problems with attached PBXs.

Some interface types do not support all of the mentioned arguments. The following table shows the supported address-complete indication conversion types and the default setting for each interface type:

Interface Type	Transparent	Set	Clear
ISDN	supported; default	supported	Supported
H.323	supported; default	supported	Supported
SIP	not supported	supported	supported; default

Call-Router

The call-router is extended to be able to set the address-complete indication or append a terminating-character in some circumstances. This includes:

- Setting the address-complete indication when the digit-collection timeout elapses
- Appending the terminating-character when the digit-collection timeout elapses
- Filtering-out the terminating-character and optionally set the address-complete indication
- Setting the address-complete indication when a called-party number matches a fully specified call-router rule ($-terminated entry).
- Appending the terminating-character when the called-party number matches a fully specified call-router rule ($-terminated entry).

The command **digit-collection timeout <secs>** has been extended with two optional arguments that specify whether a terminating-character is appended or the address-complete indication set when the digit-timeout elapses. The command syntax is

[**no**] **digit-collection timeout <secs>** [**append-terminating-char**] [**set-address-complete-indication**]

The extensions configure the action(s) to be performed when the digit-collection timeout elapses. The digit-collection timeout runs when a call is routed to a called-e164 routing-table of which a T-terminated rule is selected. Two actions are available; both can be enabled independently. The **append-terminating-char** action appends the configured terminating-character when the timeout elapses. The **set-address-complete-indication** action sets the address-complete indication. Whether or not the egress interface propagates the address-complete indication depends on the interface configuration (see below).

The following table shows the different digit-collection timeout configurations and their effect on a T-terminated route when the digit-collection timeout elapses. Important settings are marked bold.

Digit Collection Timeout Configuration			Ingress Properties		Egress Properties (Result of call-routing)	
Timeout	**append-terminating-char**	**set-address-complete-indication**	**called-e164**	**Address-Complete Indication**	**called-e164**	**Address-Complete Indication**
no	no	no	123	false	no call	
				true		
		yes		false		
				true		
	yes	no		false		
				true		
		yes		false		
				true		
yes	no	no		false	123	false
				true		true
		yes		false		**true**
				true		true
	yes	no		false	**123#**	false
				true		true
		yes		false		**true**
				true		true

The command **digit-collection terminating-char <char>** has been extended with two optional arguments that specify whether the terminating-character is re-appended or the address-complete indication is set when a digit-collection timeout is stopped by the user sending the terminating-character. The new command syntax is:

[**no**] **digit-collection terminating-char <char>** [**append-terminating-char**] [**set-address-complete-indication**]

The extensions configure what action(s) shall be performed when the digit-collection timeout is stopped by the reception of a terminating-character. Normally, without specifying an action, the received terminating-character is removed from the called-party number. The **append-terminating-char** action re-appends the terminating-character. The **set-address-complete-indication** action sets the address-complete indication. Whether or

not the egress interface propagates the address-complete indication depends on the interface configuration (see below).

The next table shows the different digit-collection terminating-character configurations and their effect on a T-terminated route when the terminating-character is received.

Digit Collection Terminating-Char Configuration			Ingress Properties		Egress Properties (Result of call-routing)	
Terminating-Char	**append-terminating-char**	**set-address-complete-indication**	**called-e164**	**Address-Complete Indication**	**called-e164**	**Address-Complete Indication**
No	no	no	123#	False	123#	false
				true		true
		yes		false		false
				true		true
	yes	no		false		false
				true		true
		yes		false		false
				true		true
Yes	no	no		false	123	false
				true		true
		yes		false		**true**
				true		true
	yes	no		false	**123#**	false
				true		true
		yes		false		**true**
				true		true

The command **digit-collection full-match** has been introduced. The syntax is:

digit-collection full-match [**append-terminating-char**] [**set-address-complete-indication**]

This command configures what action(s) shall be performed when a dollar-($)-terminated rule is selected in a called-e164 routing-table. We call such an entry a fully specified number entry. The **append-terminating-char** action appends the terminating-character; while the **set-address-complete-indication** action sets the address-complete indication. Whether or not the egress interface propagates the address-complete indication depends on the interface configuration (see below).

Egress Interface

On the egress interface (ISDN, H.323) the **address-complete-indication emit <type>** command configures how the internal representation of the address-complete indication shall be mapped to the representation of the signaling-protocol.

The possible values for the type argument are:

- **transparent**: Transparently passes an address-complete indication to the signaling-protocol (e.g. ISDN by sending a Sending Complete Information Element).
- **set**: Always sends a Sending Complete Information Element with the SETUP message. This configuration can be used to disable overlap-sending on an interface.
- **clear**: Never sends a Sending Complete Information Element. This configuration may be used in rare cases to solve interoperability problems with attached PBXs, e.g. when the attached PBX does not support the Sending Complete Information Element.

Some interface types do not support all arguments. SIP does not support this configuration at all, because SIP does not support overlap dialing. The following table shows the supported address-complete indication conversion types and the default setting for each interface type:

Interface Type	Transparent	Set	Clear
ISDN	supported; default	supported	Supported
H.323	supported; default	supported	Supported
SIP	—	—	—

The following procedure demonstrates how to disable overlap-sending for incoming SIP calls. SIP does not provide an overlap-dialing procedure; so for most applications, address-complete indications should be cleared.

Mode: context cs / interface sip

Step	Command	Purpose
1	***node*(if-sip)[*if-name*]#address-complete-indication clear**	Clear the address-complete indication for all incoming calls over this interface

The following procedure demonstrates how to create a routing-table that allows overlap dialing. When the timeout elapses or when the terminating-character is received, the address-complete indication shall be set toward the egress interface.

Mode: context cs

Step	Command	Purpose
1	***node*(cts-cs)[*ctx-name*]#routing-table called-e164 <tab-name>**	Creates a routing-table that examines the called-party number
2	***node*(rt-tab)[tab-*name*]#route T dest-interface <if-name>**	Routes any number (after the waiting for digits) to the egress interface

Step	Command	Purpose
3	***node*(rt-tab)[tab-*name*]#exit**	Leaves the routing table configuration mode and returns to the context cs configuration mode
4	***node*(cts-cs)[*ctx-name*]#digit-collection timeout 5 set-address-complete-indication**	Configures the digit-collection timeout to 5 seconds and sets the address-complete indication if the timeout elapses
5	***node*(cts-cs)[*ctx-name*]#digit-collection terminating-char # set-address-complete-indication**	Configures the terminating-character that can stop the digit-collection timeout and immediately place the call to '#' and sets the address-complete indication if the character is received

Creating call services

Routing tables, mapping tables and complex functions only manipulate address properties of a call (like the called party number). A call service is another call router entity that actively accesses the state of a call. A call service is able to spawn other calls or merge existing calls together. This allows building services like hunt groups etc.

You can view a call service as a virtual endpoint that accepts a call and creates new calls to other destinations. A call that is routed to a call service is not served by a human being but by a machine that accepts the call and performs needed actions.

The hunt group service, for example, accepts a call that is routed to it and spawns a second call that is placed to the first final destination. If this destination is not reachable, another destination is tried until one of the configured destinations accept the call.

Creating a hunt group service

A hunt group service hunts an incoming call to multiple interfaces. Figure 66 shows an example scenario where a call from a SIP interface is first processed by several tables. The second table decides that the call must be forwarded to the PSTN. The device is connected to the PSTN over four BRIs, which are bound to the context CS

interfaces IF-BRI0 up to IF-BRI3. All four ISDN interfaces lead to the same provider. Since the call router does not know the load on the BRIs, it has to be able to try BRI0 and, if BRI0 already serves two calls, use BRI1, and so on.

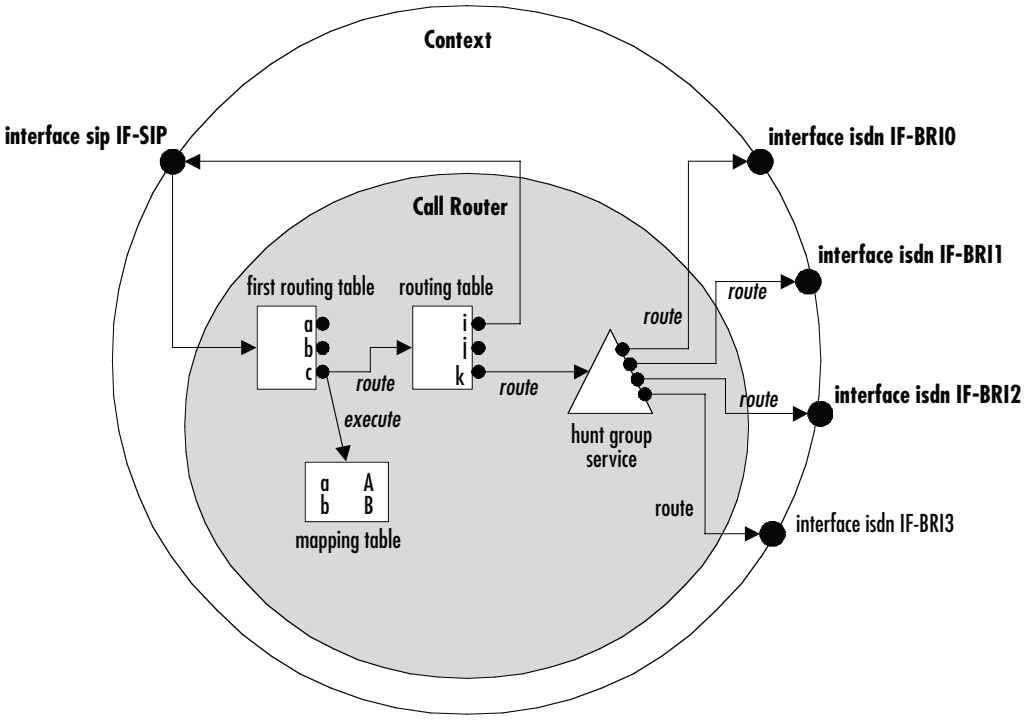


Figure 66. Hunt group service

The hunt group service accepts a call routed to it by a routing table or directly from an interface and creates another call that is offered to one of the configured destination interfaces.

The interface tried first (IF-BRI0) may drop the call telling the service that the interface has no resources to handle the call (e.g. no circuit channel available). Note that only the call between the hunt group and the destination interface is dropped, while the original call between the SIP interface and the hunt group service remains connected.

The hunt group then decides to try the next destination (IF-BRI1), which in turn also drops the call due to unavailable resources. In our example the hunt group then tries the third and eventually the fourth destination. When an interface accepts a call, the interface hunting is complete and the hunt group service merges the original with the new call to the interface that accepted the call.

You can influence the algorithm of the hunt group by several configuration commands. You can specify whether the hunt group shall always start with the same destination interface or whether it shall immediately try the next one in a round-robin fashion. This is called cyclic operation mode.

You can specify a timeout after which the next destination interface is tried when there is no answer at all from the destination interface.

You can specify drop causes that trigger hunting for the next destination. All other causes (e.g. user busy) will drop the original call.

Note Unlike previous versions of SmartWare, now you can hunt a call over different interface types, not only over ISDN interfaces. You can, e.g. create a hunt group to try to call over a H.323 interface and, if this call fails, do a fallback to an ISDN interface.

Procedure: To create and configure a hunt group service

Mode: Context CS

Step	Command	Purpose
1	*node*(ctx-cs)[*switch*]#**service hunt-group** *service-name*	Creates a new hunt group service and enters hunt group configuration mode.
2	*node*(svc-hunt)[service-name]#**cyclic** *or* *node*(svc-hunt)[service-name]#**no cyclic**	Configure the hunt group for cyclic operation mode. Subsequent calls try another first destination in a round-robin method. Default is not to use cyclic mode – always to start with the first configured destination. **Note:** When you use the hunt-group for a fallback scenario, you must switch off cyclic operation mode.
3	*node*(svc-hunt)[service-name]#**timeout** *timeout*	Configures a timeout in seconds after which the next destination is tried when the current destination does not answer at all. (Some interface (e.g. SIP) may wait an arbitrary long time until an answer is returned.) Default is not to use a timeout.
4	*node*(svc-hunt)[service-name]#**drop-cause** *cause* or *node*(svc-hunt)[service-name]#**no drop-cause** *cause*	Enables or disables another drop cause to the list. When an interface has a problem placing a call to the final destination it drops the call specifying a drop cause (e.g. user busy, no resource available). Some drop causes drop the original call while other causes trigger the hunt for another destination. This command can be used to configure the hunt behavior on destination call drop. See the list below for a summary of all available drop causes and their default state.
5	*node*(svc-hunt)[service-name]#**route call dest-interface** *interface-name* or *node*(svc-hunt)[service-name]#**route call dest-table** *table-name* or *node*(svc-hunt)[service-name]#**route call dest-service** *service-name*	Adds a route to a destination. This is the destination that is tried during hunt group's interface hunting. The destination can either be an interface or you can route the call again to a routing table or directly to another service. This allows you cascading services.
6		Repeat step 5 to add additional hunting destinations.

When a destination interface drops the call, the hunt group service has to decide whether this is because the interface is not able to handle the call (e.g. no bearer channel available) or whether the destination user does not want or is not able to communicate (e.g. user busy). In the first case, the hunt group service hunts for the next destination interface, while in the second case the original call is dropped with the same cause.

The following table lists all drop causes and specifies whether the cause is used for hunting the next destination or dropping the original call. The behavior can be configured for each hunt group individually for each cause using the **drop-cause** command in the hunt group service mode.

Table 21. Hunt group drop causes

Class	Cause	Default Behavior of the Hunt Group Service	Description
Normal Event	unallocated-number	Drop original call	The number is sent in the correct format. However, the number is not assigned to the destination equipment.
	no-route-to-network	Drop original call	The destination is asked to route the call through an unrecognized network. This cause indicates that the equipment sending this cause has received a request to route the call through a particular transit network, which it does not recognize. The equipment sending this cause does not recognize the transit network either because the transit network does not exist or because that particular network, while it does exist, does not serve the equipment that is sending this cause.
	no-route-to-destination	Drop original call	The call routes through an intermediate network that does not serve the destination address. The called user cannot be reached because the network through which the call has been routed does not serve the destination desired.
	channel-unacceptable	Drop original call	The service quality of the specified channel is insufficient to accept the connection. The call attempt failed because the channel cannot be used.
	call-awarded	Drop original call	The user assigns an incoming call that is connecting to an already established call channel.
	normal-call-clearing	Drop original call	Normal call clearing occurs. This cause indicates that the call is being cleared because one of the users involved in the call has requested that the call be cleared. Under normal situations, the source of this cause is not the network.
	user-busy	Drop original call	The called system acknowledges the connection request but cannot accept the call because all channels are in use. It is noted that the user equipment is compatible with the call.

Table 21. Hunt group drop causes (Continued)

Class	Cause	Default Behavior of the Hunt Group Service	Description
Normal Event (Cont.)	no-user-responding	Drop original call	The connection fails because the destination does not respond to the call. This cause is used when a user does not respond to a call establishment message with either an alerting or a connect indication with the prescribed period of time allocated.
	no-answer-from-user	Drop original call	The destination responds to the connection request but fails to complete the connection within the prescribed time. This cause is used when the user has provided an alerting indication but has not provided a connect indication within a prescribed period of time.
	subscriber-absent	Drop original call	The remote device you attempted to reach is unavailable and has disconnected from the network.
	call-rejected	Drop original call	The destination can accept the call but rejects it for an unknown reason. This cause indicates that the equipment sending this cause does not wish to accept this call, although it could have accepted the call because the equipment sending this cause is neither busy nor incompatible.
	number-changed	Drop original call	The number used to set up the call is not assigned to a system. This cause is returned to a calling user when the called party number indicated by the calling user is no longer assigned.
	non-selected-user-clearing	Drop original call	The destination can accept the call but rejects it because it is not assigned to the user.
	destination-out-of-order	Drop original call	The destination cannot be reached because of an interface malfunction, and a signaling message cannot be delivered. This can be a temporary condition, but it could last for an extended period.
	invalid-number-format	Drop original call	The connection fails because the destination address is presented in an unrecognizable format, or the destination address is incomplete.
	facility-rejected	Drop original call	The network cannot provide the facility requested by the user.

Table 21. Hunt group drop causes (Continued)

Class	Cause	Default Behavior of the Hunt Group Service	Description
Normal Event (Cont.)	response-to-status-enquiry	Drop original call	The status message is generated in direct response to receiving a status inquiry message.
	normal-unspecified	Drop original call	Reports the occurrence of a normal event when no standard cause applies.
Resource Unavailable	no-circuit-channel-available	Hunt for next destination	The connection fails because no appropriate channel is available to take the call.
	network-out-of-order	Hunt for next destination	The destination cannot be reached because of a network malfunction, and the condition can last for an extended period. An immediate reconnect attempt will probably fail.
	temporary-failure	Hunt for next destination	An error occurs because of a network malfunction. The problem will be resolved shortly.
	switching-equipment-congestion	hunt for next destination	The destination cannot be reached because the network switching equipment is temporary overloaded.
	access-info-discarded	Hunt for next destination	The network cannot provide the requested access information. This cause indicates that the network could not deliver access information to the remote user as requested.
	circuit-channel-not-available	Hunt for next destination	The equipment cannot provide the requested channel for an unknown reason.
	resources-unavailable	Hunt for next destination	The requested channel or service is unavailable for an unknown reason.
Service or Option Not Available	qos-unavailable	Drop original call	The network cannot provide the requested quality of service.
	facility-not-subscribed	Drop original call	The remote equipment supports the requested supplementary service by subscription only. This cause indicates that the network could not provide the requested supplementary service because the user has not completed the necessary administrative arrangements with its supporting networks.
	bearer-capability-not-authorized	Drop original call	The user requests a bearer capability the network provides, but the user is not authorized to use it. This can be a subscription problem.

Table 21. Hunt group drop causes (Continued)

Class	Cause	Default Behavior of the Hunt Group Service	Description
Service or Option Not Available (Cont.)	bearer-capability-not-available	Drop original call	The network normally provides the requested bearer capability, but it is unavailable at the present time. This can be due to a temporary network problem or subscription problem.
	service-or-option-not-available	Drop original call	The network or remote equipment cannot provide the requested service option for an unspecified reason. This can be a subscription problem.
Service or Option Not Implemented	bearer-capability-not-implemented	Drop original call	The network cannot provide the bearer capability requested by the user.
	channel-type-not-implemented	Drop original call	The network or the destination equipment does not support the requested channel type.
	facility-not-implemented	Drop original call	The remote equipment does not support the requested supplementary service.
	only-restricted-digital-available	Drop original call	The network cannot provide unrestricted digital information bearer capability. This cause indicates that a device has requested an unrestricted bearer service but the equipment sending this cause only supports the restricted version of the requested bearer capability.
	service-or-option-not-implemented	Drop original call	The network or remote equipment cannot provide the requested service option for an unspecified reason. This can be a subscription problem.
Invalid Message	invalid-call-reference	Drop original call	The remote equipment receives a call with a call reference value that is not currently in use.
	channel-does-not-exist	Drop original call	The receiving equipment is requested to use a channel that is not activated on the interface for calls. This cause indicates that the equipment sending this cause has received a request to use a channel not activated on the interface for a call.
	call-identity-does-not-exist	Drop original call	This cause indicates that a call resume has been attempted with a call identity, which differs from that in use for any presently suspended call.

Table 21. Hunt group drop causes (Continued)

Class	Cause	Default Behavior of the Hunt Group Service	Description
Invalid Message (Cont.)	call-identity-in-use	Drop original call	This cause indicates that the network has received a call suspends request. The call suspend request contained a call identity which is already in use for a suspended call within the domain of interfaces over which the call might be resumed.
	no-call-suspended	Drop original call	The network receives a call resume request when there is not a suspended call pending. This can be a transient error that will be resolved by successive call retries.
	call-has-been-cleared	Drop original call	The network receives a call resume request. This call resume request contains a call identity that once indicated a suspended call. However, the suspended call was cleared either by time-out or by the remote user.
	incompatible-destination	Drop original call	Indicates that an attempt is made to connect incompatible equipment.
	invalid-transit-network	Drop original call	This cause indicates that a transit network identification of an incorrect format was received.
	invalid-message	Drop original call	Received an invalid message with no standard cause.
Protocol Error	mandatory-ie-missing	Drop original call	The receiving equipment receives a message that does not include one of the mandatory information elements. This cause indicates that the equipment sending this cause has received a message that is missing a call property that must be present in the message before that message can be processed.
	message-type-not-implemented	Drop original call	The receiving equipment receives an unrecognized message, because the message type is invalid or the message type is valid but not supported.
	message-type-not-state-compatible	Drop original call	The remote equipment receives an invalid message with no standard cause. This cause indicates that the equipment sending this cause has received a message such that the procedures do not indicate that this is a permissible message to receive while in the current call state.

Table 21. Hunt group drop causes (Continued)

Class	Cause	Default Behavior of the Hunt Group Service	Description
Protocol Error (Cont.)	ie-does-not-exist	Drop original call	The remote equipment receives a message that includes information elements or call properties that are not recognized.
	invalid-ie-contents	Drop original call	The remote equipment receives a message that includes invalid information in the information element or call property.
	recovery-on-timer-expiry	Drop original call	Your call was not completed, probably because an error occurred.
	protocol-error	Drop original call	An unspecified protocol error with no other standard cause occurred.
Interworking	interworking	Drop original call	An event occurs, but the network does not provide causes for the action it takes. The precise problem is unknown.

Example: Create a hunt group service

This example shows how to configure the hunt group service as shown in figure 66 on page 498.

```
node(cfg)#context cs
node(ctx-cs)[switch]#service hunt-group HUNT-BRI
node(func)[HUNT-BRI]#cyclic
node(func)[HUNT-BRI]#timeout 6
node(func)[HUNT-BRI]#route dest-interface IF-BRI0
node(func)[HUNT-BRI]#route dest-interface IF-BRI1
node(func)[HUNT-BRI]#route dest-interface IF-BRI2
node(func)[HUNT-BRI]#route dest-interface IF-BRI3
```

Creating a distribution group service

A distribution group service distributes a call to multiple destinations interfaces. Figure 67 shows an example scenario where a call from a SIP interface is first processed by several tables. The second table decides that the call must be forwarded to phones that are connected to various FXS interfaces. The distribution now lets ring all the four phones at the same time.

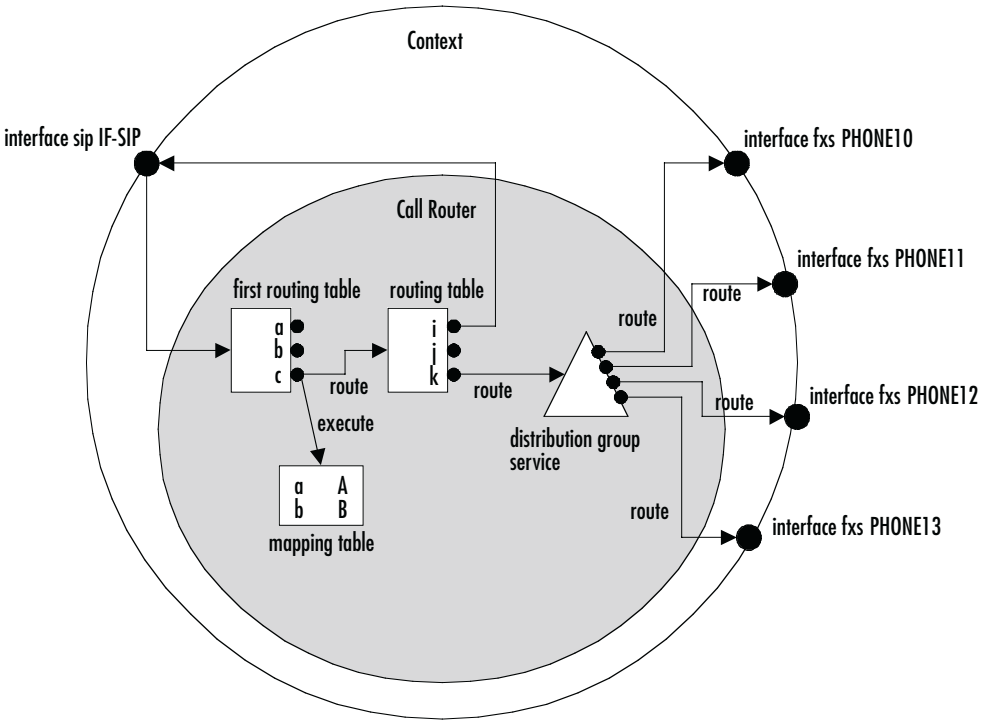


Figure 67. Distribution group service

The distribution group service accepts a call routed to it by a routing table or directly from an interface and creates four other calls that are offered to each of the configured destination interfaces.

All phones connected to the FXS interfaces (*PHONE10–PHONE13*) start ringing. Eventually one of the phones (e.g. *PHONE10*) goes off-hook. The other three calls to interfaces *PHONE11*, *PHONE12*, and *PHONE13* are immediately dropped and the phones on these interfaces stop ringing. Now the distribution service is no longer needed. Thus the service merges the original call to the accepted destination call to interface *PHONE10*.

You can configure how the distribution algorithm works in many ways. You can specify the maximum number of destination interfaces that are called at the same time. Then you can specify a timeout after which a next destination is added to the destination calls. This makes it possible to configure a scenario where a call is offered to two destinations, and (when no one answers) stop ringing the first phone but try another third destination.

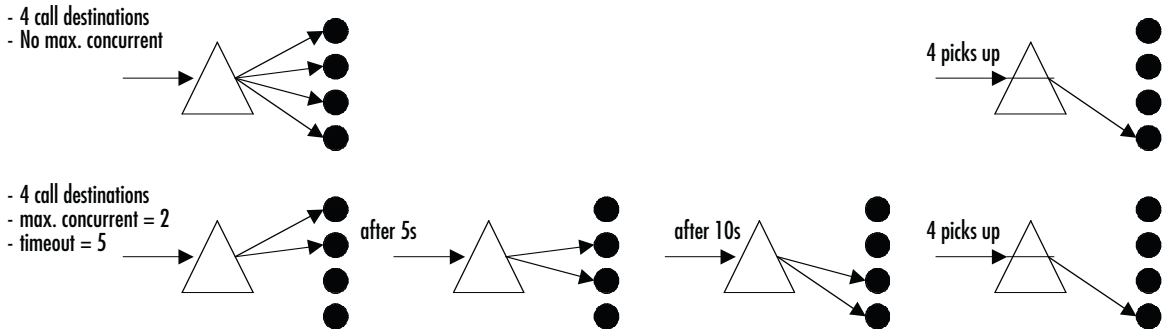


Figure 68. Distribution group service examples

Procedure: To create and configure a distribution group service

Mode: Context CS

Step	Command	Purpose
1	*node*(ctx-cs)[*switch*]#**service distribution-group** *service-name*	Creates a new distribution group service and enters distribution group configuration mode.
2	*node*(svc-hunt)[*service-name*]#**cyclic**	Configure the distribution group for cyclic operation mode. Subsequent calls try another first destination in a round-robin method. Default is not to use cyclic mode – always to start with the first configured destination(s).
3	*node*(svc-hunt)[*service-name*]#**max – concurrent** *max-concurrent*	Configures how many destinations shall be called at the same time. If you also configure a timeout, the first call is cleared and an additional call is made after that timeout. Thus only the specified number of destinations is ringing at the same time.
4	*node*(svc-hunt)[service-name]#**timeout** *timeout*	Configures a timeout in seconds after which one destination is dropped and a next destination is called.
5	*node*(svc-hunt)[service-name]#**route call dest-interface** *interface-name* or *node*(svc-hunt)[service-name]#**route call dest-table** *table-name* or *node*(svc-hunt)[service-name]#**route call dest-service** *service-name*	Adds a route to a destination. This is the interface, table or service that is tried to call during the distribution group's attempt to make calls to the destinations. The destination can either be an interface or you can route the call again to a routing table or directly to another service. This allows you cascading services.
6		Repeat step 5 to add additional hunting destinations.

Note If you specified the maximum number of concurrent destinations and the distribution group tried each destination, the final destinations ring until someone picks up one of the phones.

Note It does not make sense to configure the maximum number of concurrent destinations but no timeout, though the software does not prevent this configuration.

Distribution-Group Min-Concurrent setting

A new command in the call-control's distribution-group service lets the user specify how many of the configured call destinations should be tried first: **min-concurrent <number>**. Together with the **max-concurrent** and **timeout** commands, you can configure the behavior of the call distribution. The distribution-group starts with *min-concurrent* calls and after the *timeout* one destination call is added until *max-concurrent* is reached. Then, again after the timeout, a call to the next destination is placed while the first destination call is dropped.

Mode: service distribution-group <name>

Step	Command	Purpose
1	[*name*]**(svc-dist)**[*name*]**# min-concurrent** <number>	Sets the minimum number of concurrent calls.

Example: To configure a distribution group that first rings on the phone#1 and then, after 5 seconds, on phone#1 and phone#2, enter the following commands:

```
node(svc-hunt)[service-name]#min-concurrent 1
node(svc-hunt)[service-name]#max-concurrent 2
node(svc-hunt)[service-name]#timeout 3
node(svc-hunt)[service-name]#route call dest-interface PHONE1
node(svc-hunt)[service-name]#route call dest-interface PHONE2
```

Call-router 'limiter' service

The call-router 'limiter' service limit offers a flexible technique to limit the maximum number of concurrent calls within the system. It also limits the call-setup rate within a system. Calls exceeding the defined limits can either be simply dropped or they can be handled differently. A call is counted as an active call as soon as the call-setup message reaches the limiter service. The call remains active until the signaling has completed tearing down the call.

In the figure is a call-router configuration which uses the limiter service to limit the maximum number of concurrent calls between SIP and ISDN to 20. In this scenario, if the limit is reached, any additional call received from sip will be dropped. If however an additional call arrives from ISDN, it will be forwarded to the special ISDN interface called 'voicemail'.

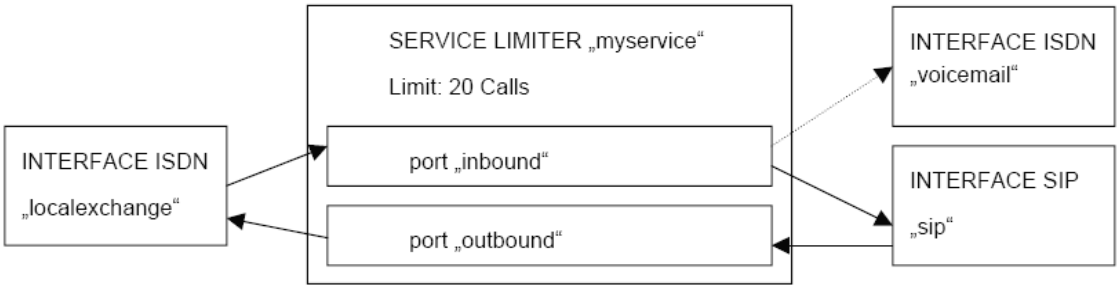


Figure 69. ‘Limiter’ service diagram

context cs switch

interface isdn localexchange

route call dest-service mylimiter.inbound

interface isdn voicemail

interface sip sip

bind gateway sip

route call dest-service mylimiter.outbound

service limiter mylimiter

max-calls 20

port inbound

route call dest-interface sip

exceed max-calls route call dest-interface voicemail

port outbound

route call dest-interface localexchange

Similarly the call-setup-rate could be limited by using the ‘exceed max-call-rate’ instead of the ‘exceed max-calls’ command in the limiter-port configuration mode. There is no limitation on the number of ports a limiter can have. You can create as many as you need for your application.

Priority service

The service ‘priority’ can automatically free resources if a high priority call needs to be established while no resources are available. The service ‘priority’ can have multiple ports. You can assign a priority level for each port. This priority level defines the priority level of each call, which is received through the port. If a call with higher priority fails to be established, the service tries dropping lower priority calls to free resources for the higher priority call. Subsequently it tries to establish the higher priority call again. Figure 70 is a typical application for this service in which non-.emergency calls are dropped to free resources for emergency calls.

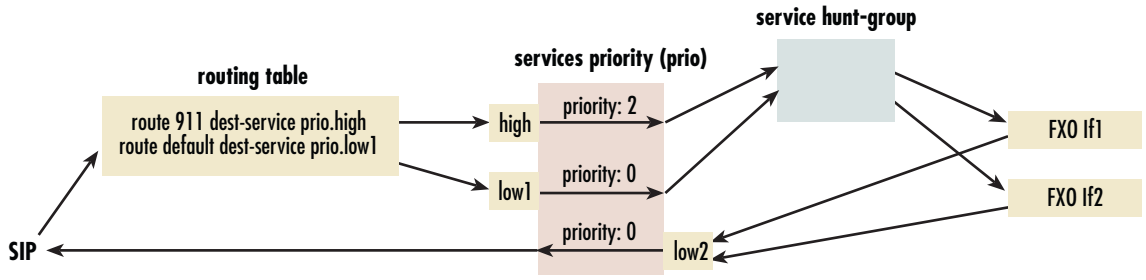


Figure 70. Priority service diagram

By default, the service drops any lower priority calls if a higher priority call fails. However, you have the option to limit the number of lower priority calls to be dropped if a higher priority call fails. It also blocks new lower priority calls for a configurable time after a higher priority call failed.

The following procedure demonstrates how to configure a priority service.

Mode: context cs

Step	Command	Purpose
1	***node*(ctx-cs)[ctx-*name*]# service priority <svc-name>**	Create a priority service.
2	***node*(svc-prio)[svc-name]# port <port-name>**	Create a port within the service.
3	***node*(port)[port-*name*]# route dest-....**	Define the routing destination for calls received on this port.
4 (Optional)	***node*(port)[port-*name*]# service priority**	Define the priority for calls received through this port. (The default priority level is 0)
5	***node*(port)[port-*name*]# exit**	Leave the port configuration mode
6		Repeat steps 2 to 5 for any additional calls you need to create.
7 (Optional)	***node*(svc-prio)[svc-name]# max-calls-to-drop <calls>**	Define the maximum number of lower priority calls to be droppedif a higher priority call fails. (Default is to drop any lower priority calls.)
8 (Optional)	***node*(svc-prio)[svc-name]# quiesce-time <seconds>**	Define the time for which lower priority calls are blocked after a higher priority call failed. (Default is 15 seconds.)
9 (Optional)	***node*(svc-prio)[svc-name]# retry-timeout <seconds>**	Define the time for which the service waits after a higher priority call failed until it tries to establish it for a second time. This allows resources used by dropped lower priority calls to get available again. (Default is 3 seconds.)

Note Although this service improves the probability that higher priority calls can be established successfully, there is no guarantee that a higher priority task can be established successfully at any time.

CS Bridge service—'VoIP Leased Line'

The circuit switch (CS) bridge service provides the functional ability to create a leased line between two FXS ports, with the FXS ports on different SmartNodes. The call is point-to-point in an *always connected* state, also known as *nailed up*. This Call Control service is called *Bridge services*."

The application for this feature is when a constantly connected VoIP link is required between two FXS ports. It is as if you connected a regular telephone that is always off-hook to the FXS port. However it is not identical since the other end of the VoIP leased-line connection does not ring like in a PLAR application. There is no end-to-end call setup so no call-progress tones are required nor needed.

Here is another way to describe the application. A user has an FXO port in two remote locations and wants to connect these two locations over an IP network. Clearly the FXO port at a location must connect to an FXS port. But since the network between the two sites is IP, there needs to be a mapping of the information on the FXS–FXO link to a method of transporting the information over an IP network. Additionally, you do not want any ringing to occur when the connection is made; you simply want it to be connected, so the SmartNodes and IP network operate transparently. See figure 71 to visualize the VoIP leased-line connection.

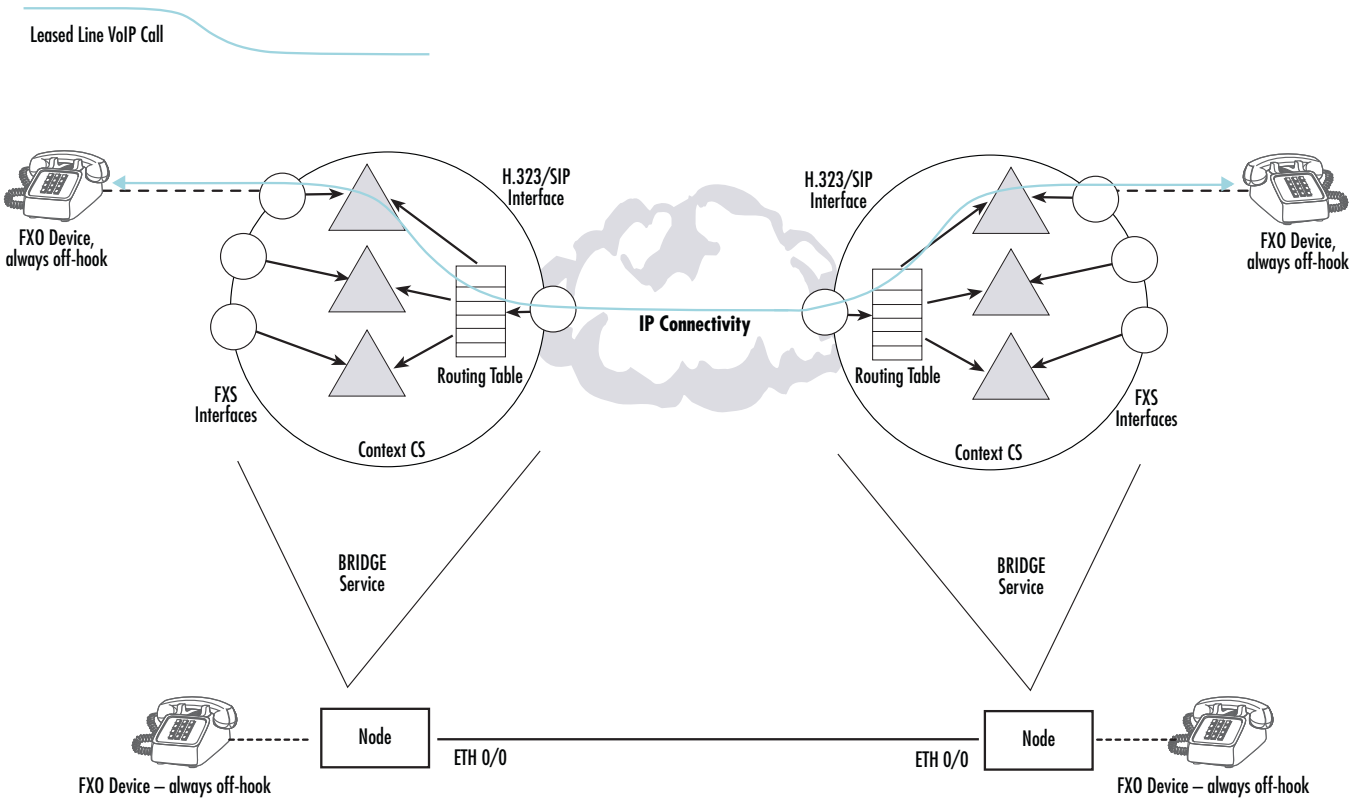


Figure 71. CS Bridge service—'VoIP Leased Line' diagram

Now we will describe the technical details and logical structure to implement this application. From the perspective of just one SmartNode, the SmartNode makes two independent calls. These two calls are made from a logical structure, called a *Bridge Service*. The Bridge Service has two interfaces, the *listener* port and the *dialer*

port (see figure 72). Each of these interfaces is responsible for one of the two independent calls. The listener port terminates the "FXS call" and the dialer port terminates the "RTP call."

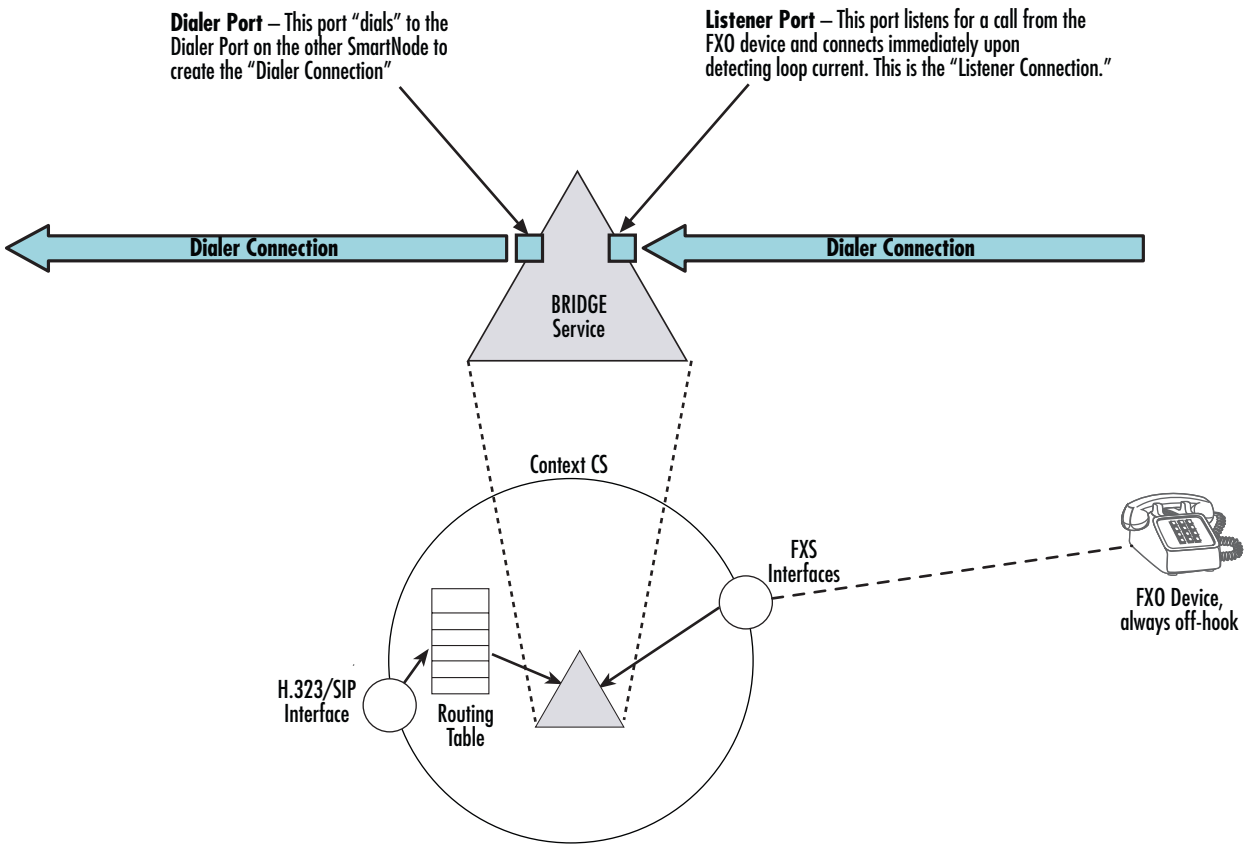


Figure 72. Bridge services diagram

The listener port interface listens to the FXS interface of the SmartNode for an active FXS–FXO connection. It recognizes an active connection by detecting current in the FXS–FXO loop, and the *FXS call* is established. The dialer port interface attempts to make and keep an RTP connection session or *call* with a dialer port in a remote SmartNode. It is called an *RTP call*. This connection is over the IP network. The listener port and the dialer port both try to keep their individual calls up and operating at all times. However if the listener port loses its connection (that is, its call), the dialer port does not disconnect its RTP call but remains connected to the other SmartNode's dialer port. Similarly, if the local dialer port loses its connection with the remote SmartNode's dialer port, the SmartNode's listener port does not disconnect its FXS call but remains connected to the FXO device. Though the calls operate independently, they operate over a single data path from end-to-end.

The dialer port is bound to the Routing Table which is subsequently bound to either an H.323 or SIP interface. The routing table makes the connection to the proper FXS interface.

Mode: Context CS

Step	Command	Purpose
1	[*name*] **(ctx-cs)[switch]#[no] service bridge BRIDGE1**	Enters the bridge service configuration mode / deletes a bridge service.
2	[*name*] **(svc-bridg)[BRIDGE1]#port DIALER**	Creates a port on the service that can accept or spawn calls (the max number of ports is currently limited to two)
3	[*name*] **(port)[DIALER]# dial persistent 123 dest-interface REMOTE**	Configures the port to actively dial the called-party "123" to the destination REMOTE. This connection is kept open until the service is shut down, or a "no dial" command is issued. If the remote side terminates the connection, the port tries to reconnect. As soon as the connection is established, the call is connected to all other calls present on the service.
4	[*name*] **(svc-bridg)[BRIDGE1]#port LISTENER**	Creates a port on the service that can accept or spawn calls (the max number of ports is currently limited to two)
5	[*name*] **(port)[DIALER]# no dial**	Without entering a "dial" command, or by specifying "no dial", this port does not dial, but listen for incoming calls. If a call comes in, it is automatically connected to all other calls present on the service.

Configuration Example:

```
context cs switch

  routing-table called-e164 DISPATCH
    route 1 dest-service BRIDGE1.dialer
    …

  service bridge BRIDGE1
    port listener
      mode listening
      no shutdown
    port dialer
      mode dial-persistent 1 dest-interface REMOTE
      no shutdown

interface h323 REMOTE
    bind gateway h323
    route call dest-table DISPATCH
    remote 172.16.32.37

  interface fxs PHONE1
    route call dest-service BRIDGE1.listener
```

Configuring the service second-dialtone

If a call is routed into the service second-dialtone, the service establishes the call and plays a dial tone to the caller.When a call is entering the service it is first routed to the configured destination. In case the call cannot be placed, the service plays the second dial-tone or the configured announcement to the caller until the caller enters the first DTMF digit.

Caveat: For calls from the IP side, the service only works if the G711.alaw codec is chosen. On the SmartNode 4960 the service is available for all configurable codecs.

Mode: Context CS

Step	Command	Purpose
1	*node*(ctx-cs)[*switch*]#**service second-dialtone *name***	Enters the service second-dialtone mode.
2	*node*(svc-2dt)[*name*]#**route call dest-interface|dest-service|dest-table *name***	Defines the call destination. If the call cannot be placed the second-dialtone is played.
3	*node*(svc-2dt)[*name*]#**route announcement dest-interface|dest-service|dest-table *name***	Instead of playing a dialtone locally, the service sets up a call to the configured message provider.
4	*node*(svc-2dt)[*name*]#**use profile tone-set *name***	Apply to use a alternative tone-set profile. If not configured the profile tone-set default will be used.

Deleting call services

To remove individual call services you can use the **no** form of the **service** command.

Procedure: To delete a call service

Mode: Context CS

Step	Command	Purpose
1	*node*(ctx-cs)[*switch*]#**no service** *service-name*	Delete the service *service-name*. **Note:** You do not have to enter the type of the service when just deleting it. The type must only be specified when creating a service.

Example: Remove an entire mapping table

```
node(cfg)#context cs
node(ctx-cs)[switch]#no service HUNT-BRI
```

Activate the call router configuration

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

The call router configuration is activated as soon as the CS context comes out of shutdown (e.g. at boot time or by manually entering command **no shutdown**). You can modify the configuration at runtime; changes will be active after 3 seconds. SmartWare offers a number of possibilities to monitor and debug the CS context and call router configurations. For more information refer to chapter 52, "VoIP debugging" on page 624.

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

Procedure: To show and activate the call router configuration

Mode: Context CS

Step	Command	Purpose
1	*node*(ctx-cs)[*switch*]#**show call-router config**	Show the actual call router configuration. This displays all routing and mapping tables in the current context CS. When you are inside a routing or mapping table configuration mode, only the current table is displayed.
2	*node*(ctx-cs)[*switch*]#**show running-config**	Show the whole running config includes the call routing tables
3	*node*(ctx-cs)[*switch*]#**debug call-router detail** *level*	Enable the call router debug monitor. Use level 1 for get informed about errors and increase the level up to 5 to track calls during route lookups.
4	*node*(ctx-cs)[*switch*]#**no shutdown**	Activate the whole CS context configuration including the call router configuration.
5	*node*(ctx-cs)[*switch*]#**show call-router status**	Show the actual call router status. This command can be used to examine whether or not the call router accepted all routing entries as entered in the configuration.

Note Unlike previous versions of SmartWare you must explicitly enter the **no shutdown** command to activate the call router.

Test the call router configuration

After activating the call router configuration you can test the call router by simulating a route lookup as if a call is routed to a table. You have to execute the test call-router command and specify all necessary call properties together with the routing table you want to test.

Note You must activate the call router using the **no shutdown** command first.

Procedure: To test the call router configuration

Mode: Context CS

Step	Command	Purpose
1	*node*(ctx-cs)[*switch*]#**debug call-router detail *level***	Enables the call router debug monitor. Chose level 5 to trace route lookups in detail.
2	*node*(ctx-cs)[*switch*]#**test call-router** *table-name* **[***property-type property-value***]**	Tests the routing or mapping table *table-name* with the specified call property. You can repeat the optional section multiple times and thus enter as many call properties as you want.

Example: Create and test a routing table

```
node(cfg)#context cs
node(cts-cs)[switch]#routing-table called-e164 TEST
node(rt-tab)[TEST]#route 1 dest-interface IF1
node(rt-tab)[TEST]#route 1[0-4] dest-interface IF2
node(rt-tab)[TEST]#route 11 dest-interface IF3
node(rt-tab)[TEST]#route 111T dest-interface IF4
```

```
node(rt-tab)[TEST]#route default dest-interface IF5
node(rt-tab)[TEST]#exit
node(ctx-cs)[switch]#no shutdown
node(ctx-cs)[switch]#debug call-router detail 5
node(ctx-cs)[switch]#test call-router TEST called-e164 123
Parameters
==========

 Time:                                  2004-03-02T16:55:33<-- Time of the lookup
 Result:                                route-found-place-call<-- Lookup result
 Destination:                           IF2<-- Dest. Interface
 Timeout:                               0<-- Digit-Coll. TO

 Property Containers
 -------------------

  Properties

  Properties
   E164-Number:                         123 (String)<-- CdPN after lookup/change

  Properties

16:55:33  CR    > [switch] Routing-Lookup:
16:55:33  CR    >   Find best-matching called-element in table test
16:55:33  CR    >     01: Prefix Timeout Expression: E164-Number of 123 completely
(no timeout) matches 1
16:55:33  CR    >     02: Prefix Timeout Expression: E164-Number of 123 completely
(no timeout) matches 1[0-4]
16:55:33  CR    >     03: Prefix Timeout Expression: E164-Number of 123 does not
match 11
16:55:33  CR    >     04: Prefix Timeout Expression: E164-Number of 123 does not
match 111
16:55:33  CR    >     Selecting entry 2
16:55:33  CR    >   Execute all elements in table IF2
16:55:33  CR    >   Execute all elements in table route-found-place-call
16:55:33  CR    >   Lookup result: Route found; place call (timeout=0)
```

Example: Enterprise network with local breakout and IP carrier access

Consider the following Enterprise Network.

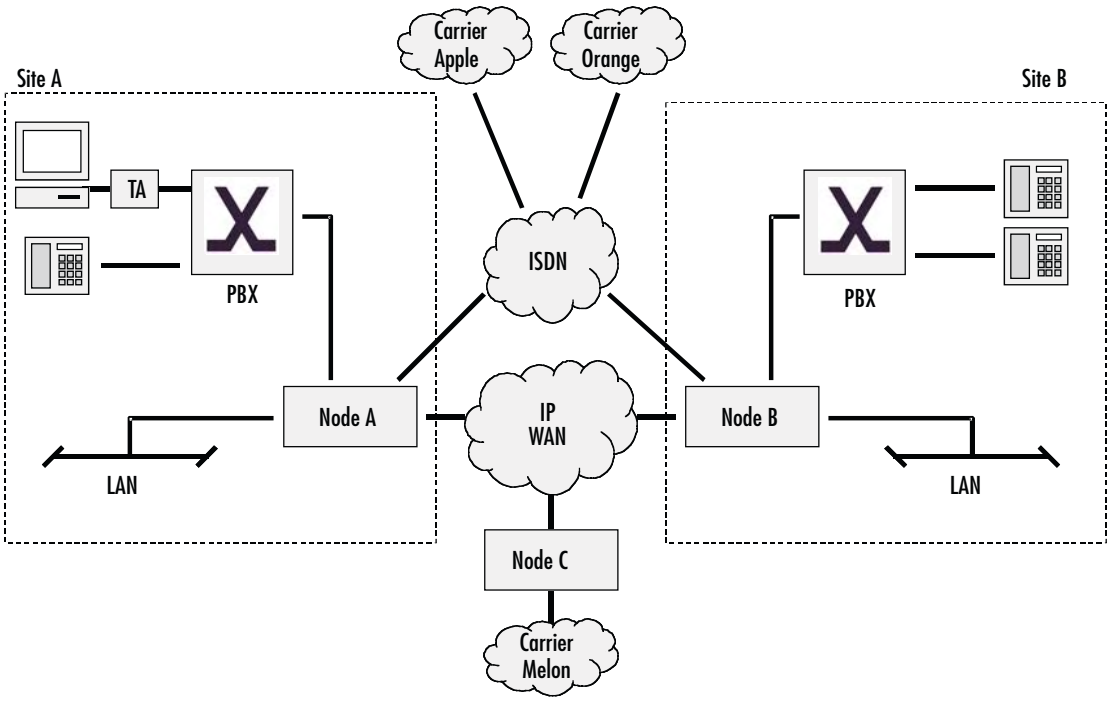


Figure 73. Call routing example network

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

The goal of this scenario is to connect the two PBX of sites *A* and *B*. The two sites are connected to a broadband IP provider. The IP network is used to exchange data and voice calls between the two sites. On the IP network there is also a PSTN gateway (Node *C*) to an alternative voice carrier *Melon* that shall be used for most call destinations.

Sites A and B also have connections to the local ISDN network. This is called the local breakout connection. The local breakout is to be used as a fallback for ISDN data connections.

We assume the following:

- The number block for site A is 022 782 55 00 to 99
- The number block for site B is 033 665 2 000 to 999
- The Carrier Access Code (CAC) for *Apple* is 1055
- The Carrier Access Code (CAC) for *Orange* is 1066
- Carriers Apple, Orange and Melon do not support ISDN data calls (PC with ISDN Terminal Adapter behind PBX A)

- When calling through carrier *Melon* the CLI (calling party number) must not use the public number blocks of Site A and B
- Carrier *Orange* is to be used for national calls
- Carrier *Apple* is to be used for calls to mobile

The requirements for the call router can be summarized as:

1. Route ISDN data calls to the local breakout.
2. Route inter-site calls to the opposite SmartNode (node A to node B and vice versa).
3. Route international calls to carrier *Melon.*
4. Provide a fallback for all VoIP calls on the local breakout.
5. Route local calls to the local breakout.
6. Route national calls to carrier *Orange.*
7. Route mobile calls to carrier *Apple.*
8. Calls from the PSTN, nodes B and C are forwarded directly to the PBX.

The remainder of this example will focus on the configuration for Node A. The configuration for Node B can be built accordingly. Node C has an even simpler configuration.

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

- Bearer capability table named *TAB-ISDN-SERVICE*, needed for requirement 1.
- Called party number table named *TAB-DEST-A*, needed for requirements 2, 3, 6 and 7
- CAC insertion for *Apple MAP-CAC-APPLE*, needed to add a carrier access code for *Apple*
- CAC insertion for *Orange MAP-CAC-ORANGE*, needed to add carrier access code for *Orange*
- CLI replacement for *Melon MAP-CLI-MELON*, needed to add carrier access code for *Melon*
- PSTN interfaces *IF-PBX-A* and *IF-LOCAL-BREAKOUT*, needed for requirements 4, 5 and 8.
- H.323 interface *IF-NODE-B*, needed for requirement 2.
- SIP interface *IF-NODE-C*, needed for requirement 3.

Figure 74 shows the corresponding CS Context and call router elements in node A:

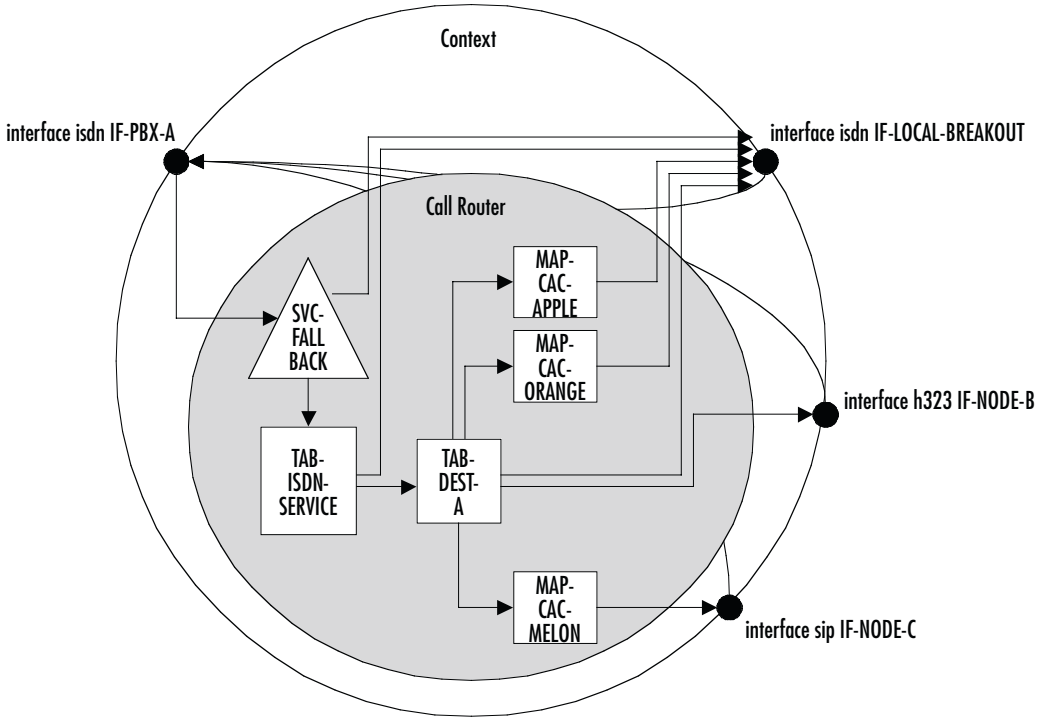


Figure 74. CS context and call router elements

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

Since the command sequence is quite long it is useful to create the configuration offline and download it using TFTP.

Note In the following lines the prompt is omitted as in a configuration file and for better readability.

```
#------------------------------------------------------------
# Call Router Config File
#------------------------------------------------------------

context cs switch

#
# Hunt-group service "SVC-FALLBACK" to catch VoIP network errors
#

service hunt-group SVC-FALLBACK
  no cyclic
  timeout 6
  route call 1 dest-table ISDN-SERVICE
  route call 2 dest-interface LOCAL-BREAKOUT
```

```
#
# Bearer capability routing table “TAB-ISDN-SERVICE”
#

routing-table itc TAB-ISDN-SERVICE
  route unrestricted-digital dest-interface IF-LOCAL-BREAKOUT
  route default dest-table TAB-DEST-A

#
# Called party number routing table “TAB-DEST-A”
#

routing-table called-e164 TAB-DEST-A
  route 0 dest-interface IF-LOCAL-BREAKOUT MAP-CAC-ORANGE
  route 00 dest-interface IF-NODE-C MAP-CLI-MELON
  route 07[4-6] dest-interface IF-LOCAL-BREAKOUT MAP-CAC-APPLE
  route 0336652... dest-interface IF-NODE-B
  route default dest-interface IF-LOCAL-BREAKOUT

#
# Number manipulation “CAC-APPLE”; add prefix 1055
#

mapping-table called-e164 to called-e164 MAP-CAC-APPLE
  map (.%) to 1055\1

#
# Number manipulation “CAC-ORANGE”; add prefix 1066
#

mapping-table called-e164 to called-e164 MAP-CAC-ORANGE
  map (.%) to 1066\1

#
# Number manipulation “CLI-MELON”
# Truncate CLI to last 2 digits and add 08004455 prefix in front
#

mapping-table calling-e164 to calling-e164 MAP-CLI-MELON
  map .%(..) to 08004455\1
```

Prior to downloading this file you should make sure there are no other tables and functions in the call router.

```
node(ctx-cs)[switch]#copy tftp://172.16.36.20/configs/SRconf.cfg running-config
Download...100%
```

Now we have to enable advanced call routing for outgoing calls and basic interface routing for incoming calls: Calls arriving on the interface from the PBX are routed to the SVC-FALLBACK service while incoming calls from the other interfaces are routed directly to the IF-PBX-A interface.

```
node(ctx-cs)[switch]#interface isdn IF-PBX-A
node(if-pstn)[IF-PBX-A]#route dest-service SVC-FALLBACK
node(if-pstn)[IF-PBX-A]#exit
node(ctx-cs)[switch]#interface isdn IF-LOCAL-BREAKOUT
node(if-isdn)[IF-LOCA~]#route dest-interface IF-PBX-A
```

```
node(if-isdn)[IF-LOCA~]#exit
node(ctx-cs)[switch]#interface h323 IF-NODE-B
node(if-h323)[IF-NODE-B]#route dest-interface IF-PBX-A
node(if-h323)[IF-NODE-B]#exit
node(ctx-cs)[switch]#interface sip IF-NODE-C
node(IF-NODE-C)[IF-NODE-C]#route dest-interface IF-PBX-A
node(IF-NODE-C)[IF-NODE-C]#exit
```

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

```
node(cfg)#debug call-router
node(cfg)#context cs
node(ctx-cs)[switch]#no shutdown
02:14:30  CR    > Updating tables in 3 seconds...
02:14:33  CR    > [switch] Reloading tables now
02:14:33  CR    > [switch] Flushing all tables
02:14:33  CR    > [switch] Loading table 'TAB-ISDN-SERVICE'
02:14:33  CR    > [switch] Loading table 'TAB-DEST-A'
02:14:33  CR    > [switch] Loading table 'CAC-APPLE'
02:14:33  CR    > [switch] Loading table 'CAC-ORANGE'
02:14:33  CR    > [switch] Loading table 'CLI-MELON'
02:14:33  CR    > [switch] Loading table 'MAP-CAC-APPLE'
02:14:33  CR    > [switch] Loading table 'MAP-CAC-ORANGE'
02:14:33  CR    > [switch] Loading table 'MAP-CLI-MELON'
02:14:33  CR    > [switch] Loading table 'IF-LOCAL-BREAKOUT-precall-service'
02:14:33  CR    > [switch] Loading table 'IF-PBX-A-precall-service'
02:14:33  CR    > [switch] Loading table 'IF-NODE-B-precall-service'
02:14:33  CR    > [switch] Loading table 'IF-NODE-C-precall-service'
node(ctx-cs)[switch]#
```

Configure partial rerouting

To save bandwidth and B-Channels, SmartWare supports partial rerouting. SmartWare accepts rerouting requests from PBX, which want to push-back a diverted or forwarded call with a Rerouting Request. SmartWare can generate such a reroute request to push-back a looped call. Both, acceptation and emission of rerouting requests can be configured separately.

To prevent rerouting when a particular service is invoked in a call, services can be configured to allow or to forbid rerouting of calls of the service. In order that a call can be rerouted, all services participating in this call must allow rerouting. In addition rerouting emit must be configured on the interface where this call comes in and is routed out.

To accept or emit rerouting requests, see “Call reroute” on page 522.

To allow push-back, see “Allow Push-Back” on page 522.

Call reroute

The **call-reroute** commands enable acceptation and emission of rerouting requests.

Enable acceptation of rerouting requests on ISDN. If a reroute is accepted, the participant who sends the reroute request is disconnected and the call is established from the SmartNode to the new destination.

Mode: context cs/interface isdn

Step	Command	Purpose
1	[*name*]**(if-isdn)**[*interface*]**#[no] call-reroute accept**	Enables acceptation of rerouting requests from ISDN. Default is disabled.

Enable emission of rerouting requests on ISDN. To reroute a call must enter and leave SmartNode through the same ISDN interface and every service invoked must allow push-back.

Mode: context cs/interface isdn

Step	Command	Purpose
1	[*name*]**(if-isdn)**[*interface*]**#[no] call-reroute emit**	Enables emission of rerouting requests from ISDN. Default is disabled.

Enable sending of "302 moved temporary" message on SIP. To reroute a call must enter and leave SmartNode through the same SIP gateway and every service invoked must allow push-back.

Mode: context cs/interface sip

Step	Command	Purpose
1	[*name*]**(if-sip)**[*interface*]**#[no] call-reroute emit**	Enables emission of "302 moved temporarily" message on SIP. Default is disabled.

Allow Push-Back

The **push-back** command allows or forbids rerouting of a call, if the service is invoked.

Enable push-back – aaa service.

Mode: context cs/service aaa

Step	Command	Purpose
1	[*name*]**(svc-aaa)**[*service*]**#[no] allows-push-back**	Enables push-back of a call of this service. Default is disabled.

Enable push-back – bridge service.

Mode: context cs/service bridge

Step	Command	Purpose
1	[*name*]**(svc-brdg)**[*service*]**#[no] allows-push-back**	Enables push-back of a call of this service. Default is disabled.

Enable push-back – distribution-group service.

Mode: context cs/service distribution-group

Step	Command	Purpose
1	[*name*]**(svc-dist)**[*service*]**#[no] allows-push-back**	Enables push-back of a call of this service. Default is disabled.

Enable push-back – hunt group service.

Mode: context cs/service hunt-group

Step	Command	Purpose
1	[*name*]**(svc-hunt)**[*service*]**#[no] allows-push-back**	Enables push-back of a call of this service. Default is disabled.

Enable push-back – limiter service.

Mode: context cs/service limiter

Step	Command	Purpose
1	[*name*]**(svc-lim)**[*service*]**#[no] allows-push-back**	Enables push-back of a call of this service. Default is enabled.

Enable push-back – priority service.

Mode: context cs/service priority

Step	Command	Purpose
1	[*name*]**(svc-prio)**[*service*]**#[no] allows-push-back**	Enables push-back of a call of this service. Default is enabled.

Chapter 41 SIP call-router services

Chapter contents

Introduction 525
SIP conference-service 525
SIP conference-service configuration task list 525
Entering conference-service configuration mode 525
Configuring the call routing destination 525
Configuring the conference server 526
SIP location-service 526
SIP location-service configuration task list 527
Entering SIP location-service configuration mode 527
Binding a location service 528
Configuring multi-contact behavior 528
Configuring the hunt timeout 528

Introduction

This chapter contains the description of all SIP specific call router services, which are only available if the software includes the SIP component.

SIP conference-service

RFC4240 describes how to address different services on a media server without additional SIP headers or header parameters. This mechanism makes use of the fact media servers do not manage users, they manage media services. For that reason RFC4240 defines how to use the user part of the Request-URI as a service indictor. The conference-service provides the functionality described in RF4240 as 'Conference Service'. It builds the Request-URI to address a conference room instance on a media server according to this standard. The user part of the Request URI will start with the service indicator 'conf=' followed by a unique conference room identifier. All calls with the same conference room identifier will attend the same session. The host part of the Request-URI that represents the media server host name has to be configured by the user. The conference room identifier and the called party number will be taken concatenated with the serial number of the device.

This looks as follows: conf=<called-nbr><serial-nbr>@<media-server>

SIP conference-service configuration task list

The following section describes how to create a new conference-service and how to enter the configuration mode of an existing one. In addition it describes all commands and sub commands of the conference-service configuration mode. All configuration tasks for a conference-service are listed below.

- Enter conference-service configuration mode (see page 525)
- Configure the call routing destination (see page 525)
- Configure the conference server (see page 526)

Entering conference-service configuration mode

The **service sip-conference** command enters the configuration mode of an existing conference-service or creates a new one with a specified name. It also destroys an existing service by using the no form of the command.

Mode: Context CS

Step	Command	Purpose
1	**[name] (ctx-cs)[switch]#[no] service sip-conference** *<service-name>*	Creates/Destroys a SIP conference-service or enters the configuration mode of an existing one.

Configuring the call routing destination

The call routing destination specifies the next interface, table or service to which the current call has to be forwarded. Detailed information about call routing of a call control service can be found in Chapter 40, "Call router configuration" on page 456. Because this is a SIP specific service, the user should be aware that the final destination of the configured call routing must be a SIP interface. This SIP interface will then set up the call to the configured conference server. Be aware that the elements in the routing path do not modify the Request-URI because it has been built by this service. Therefore the final SIP interface must not have configured a remote host.

Mode: Service SIP conference

Step	Command	Purpose
1	**[name] (svc-sip-conf)[name]#[no] route call dest-interface** *<name>* or **dest-table** *<name>* or **dest-service** *<name>*	Specifies the next call routing destination for an incoming call. The no form of the command deletes the current routing entry.

Configuring the conference server

As described in the introduction, one of the responsibilities of this service is to build the conference server Request-URI according to RFC4240. The user part will be built from the called-e164 property and the serial number whereas the host part is specified by this command.

Mode: Service SIP conference

Step	Command	Purpose
1	**[name] (svc-sip-conf)[name]#[no] sconference-server** *<host-name>* **[***<port>***]**	Specifies the conference server host name and the port. The no form of the command removes the current entry.

SIP location-service

Note To avoid possible name conflicts, two expressions will be declared here.
Location Service: Domain based identity data base.
Service Location-Service: A call-route service that accesses the Location Service declared above.

This service is the main consumer of the address bindings (mapping of the users identity to a contact address) deposited in the location service data base (see 51, "Location Service" on page 607). Address bindings have been entered to location service data base either by inbound registration or manually by the user. An address binding is an entry that describes on which host a user is currently reachable. If a call is routed to this service, it performs a lookup with the requested URI to find the right contact information.

If the call is originated by the PSTN network, then this URI does not yet exist and must be built first with a mapping table. If this is not done, the lookup uses the called-e164 number and checks if a registered identity matches this number. For more information about this process, see section, "B2B User Agent with Registered Clients" on page 571.

It is also possible a user is registered with more than one contact. This can be if the user registers with its office SIP phone and also with its SIP soft client. In this case, the location-service's behavior can be configured. On the location-service, no routing command is available. The call will automatically be routed to that SIP interface on which the register request has been received, therefore manually entered address bindings must be provided with a SIP interface as routing destination.

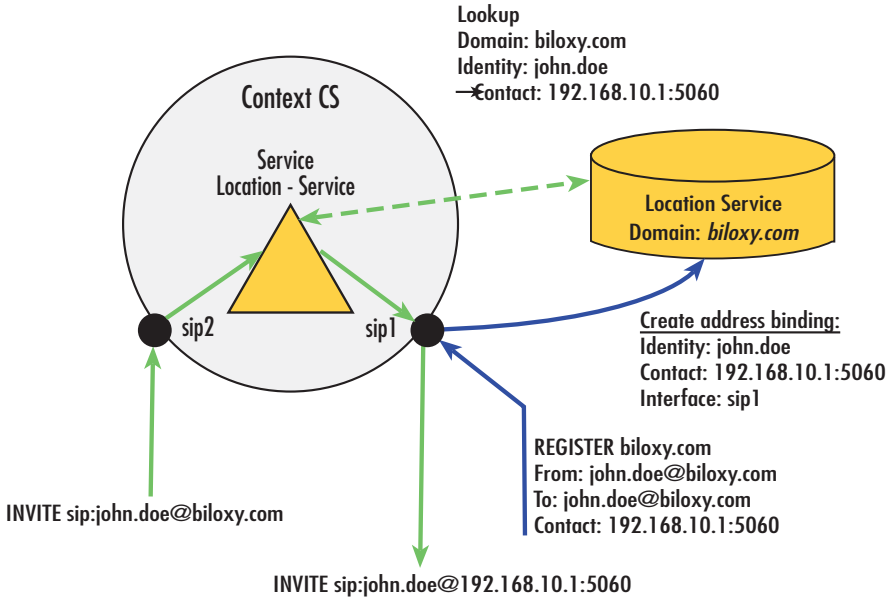


Figure 75. Registration and Lookup

SIP location-service configuration task list

The following section describes how to create a new location-service and how to enter the configuration mode of an existing one. In addition, it describes all commands and sub commands of the location-service configuration mode. All configuration tasks for a location-service are listed below.

- Enter location-service configuration mode (see page 527)
- Bind a location service (see page 528)
- Configure multi contact behavior (see page 528)
- Configure the hunt timeout (see page 528)

Entering SIP location-service configuration mode

The **service sip-location-service** command enters the configuration mode of an existing location-service or creates a new one with a specified name. It also destroys an existing service by using the no form of the command.

Mode: Context CS

Step	Command	Purpose
1	**[name] (ctx-cs)[switch]#[no] service sip-location-service** *<service-name>*	Creates/Destroys a SIP location-service or enters the configuration mode of an existing one.

Binding a location service

If a call is route to the location-service it performs a lookup with the requested URI to the bound location service to find the address bindings. This command is optional because if no location service is bound, all existing location services will be considered to find the right contact information.

Mode: Service SIP location-service

Step	Command	Purpose
1	**[name] (svc-ls)[name]#[no] bind location-service** *name*	Binds the service to a location service. The no form of the command removes an existing binding.

Configuring multi-contact behavior

There may be cases when a user has registered with several contacts in the location service database to be reachable on several SIP clients. If a call is routed to this service and the lookup in the location service database results in several contacts, this command specifies the behavior for such a case. One possibility is to contact all clients at the same time (distribute) or to contact the clients one after the other (hunt). At time of registration a priority value can be provided. In hunt mode, this priority defines the sequence of contacting the clients. The following modes are available:

- **Hunt:** Contact one registration after the other, depending on the priority. Highest priority first.
- **Distribute:** Contact all registrations at the same time.
- **Distribute and Hunt**: Registrations with the same priority build a distribution group. Hunt over this groups beginning with the highest priority.

Mode: Service SIP location-service

Step	Command	Purpose
1	**[name] (svc-ls)[name]#mode {hunt | distribute | distribute-and-hunt}**	Specifies the multi contact behavior of the SIP location-service. *Default:* distribute-and-hunt

Configuring the hunt timeout

The **hunt-timeout** command is used in 'hunt' mode and 'distribute-and-hunt' mode. It specifies the time that the service must hunt to the next contact if one is not responding. The user must configure this value because it is set per default to zero. This means that the hunting process will stay on the first contact and will never proceed with the next registered contact.

Mode: Service SIP location-service

Step	Command	Purpose
1	**[name] (svc-ls)[name]#[no] hunt-timeout** *<seconds>*	Defines the seconds to wait before hunting to the next contact or group of contacts.

Chapter 42 Tone configuration

Chapter contents

Introduction 530
Tone-set profiles 530
Tone configuration task list 531
Configuring call-progress-tone profiles 531
Configure tone-set profiles 532
Enable tone-set profile 533
Show call-progress-tone and tone-set profiles 534

Introduction

This chapter gives an overview of call-progress-tone profiles and tone-set profiles, and describes the tasks involved in their configuration.

In-band tones keep the user informed about the state of his call or additional services such as call-waiting, hold etc. Other tones can be assigned to any event that occurs during a call, a call waiting tone, for example. The in-band tones are referred to as *call-progress-tones*.

Tone-set profiles

In traditional PSTN networks the in-band tones (dial tone, alerting tone, busy tone etc.) are generated by the network, i.e. the Central Office switch or a similar device, and are relayed transparently by the SmartNode. In voice over IP networks however this model of a network side providing services including in-band tones is not given in all situations. For example, two SmartNodes may be connected directly to each other over the access network without the intervention of a traditional Central Office switch. This imposes the need to generate the local in-band tones directly on the gateways since none of the attached ISDN devices (PBXs, phones) will do so itself (ISDN USR side). The in-band tones that can be generated by the SmartNode are the following:

- Busy tone—Tone you hear when you try to reach a remote extension but it is busy.
- Confirmation tone—Tone you hear when you enable a supplementary service and the system has accepted and activated it (for future use).
- Congestion tone—Tone you hear when you try to reach a remote extension but the network is busy or out of order (for future use).
- Dial tone—Tone you hear when you lift the handset and the network is ready to accept the dialed digits of the called party number.
- Hold tone—Tone you hear when you are in an active connection and the remote extension sets you ‘On Hold’ to reach a third party extension.
- Release tone—Tone you hear when you are in an active connection and the remote extension terminates the call.
- Ringback tone—Tone you hear when the called party number is complete and the remote extension is ringing.
- Special dial tone—Tone you hear when you lift the handset and the network is ready to accept the dialed digits of the called party number but on your system is still a supplementary service activated (for future use).
- Special Information tone—Tone you hear when you try to reach a nonexistent remote extension (for future use).
- Waiting tone—Tone you hear when you already have an active connection and a second new extension tries to reach you.

All call-progress-tones are collected in a tone-set profile. A tone-set profile collects typically all the required tones for one country. The tone-set profile is assigned to the PSTN interface (ISDN, FXS, FXO) or if it is required to have different tones for individual PSTN interfaces it’s possible to assign for each PSTN interface its own tone-set profile. If no tone-set is assigned to a PSTN interface the default tone-set is taken. Figure 76 illustrates the relation ship between call-progress-tone profiles, tone-set profiles and PSTN interfaces.

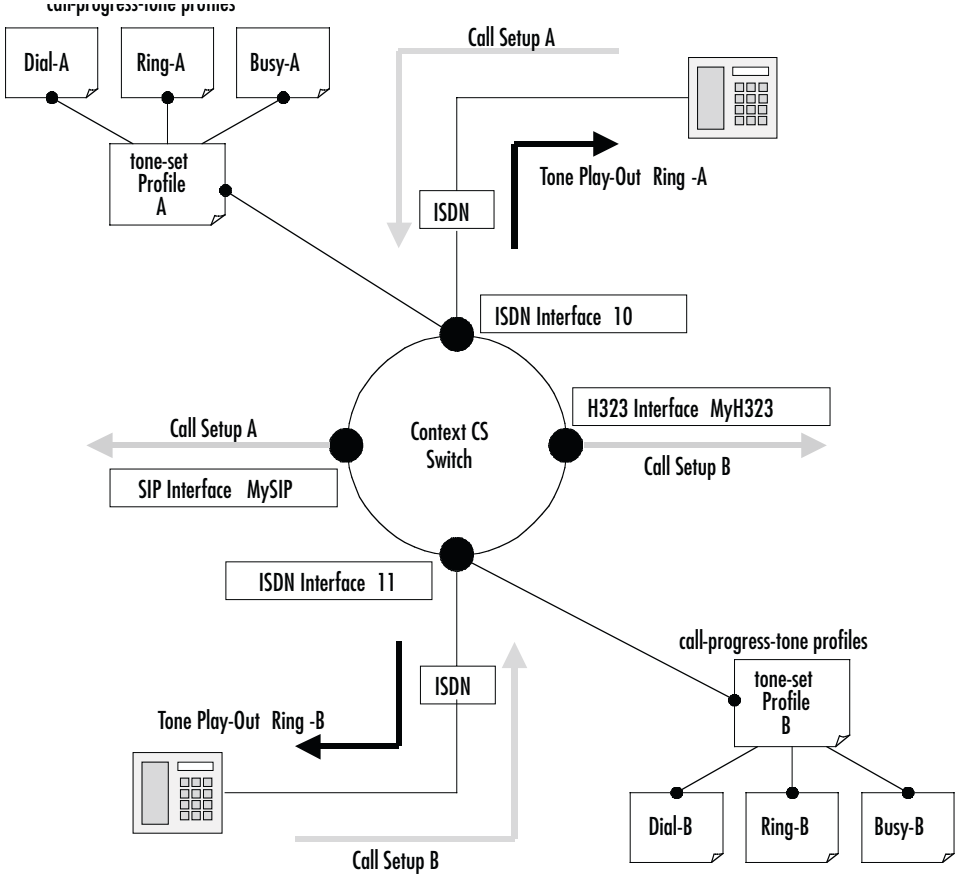


Figure 76. Assign tone-sets to a PSTN interfaces

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

Tone configuration task list

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

- Configuring call-progress-tone profiles
- Configuring tone-set profiles
- Enabling the generation of local in-band tones
- Showing call-progress-tone and tone-set profiles

Configuring call-progress-tone profiles

Each call-progress-tone consists of a sequence of different tones and pauses. Arbitrary tone cadences can be configured. With these parameters all country specific tones can be defined. Tone configuration knows only one command that have to be used repeatedly. The sequence in which the commands are entered (or appear in the config file) defines the sequence in which the corresponding elements are played.

Procedure: To configure a tone-set profile

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#profile call-progress-tone** *name*	Creates a call-progress-tone profile with name *name* and enters call-progress-tone configuration mode.
2	***node*(pf-callp)[*name*]#play** *duration frequency1 level1 [frequency2 level2]*	Defines a tone with duration *duration*, frequency *frequency1* and volume *level1*. If a second frequency is defined both frequencies are played in parallel and for the same duration
3	***node*(pf-callp)[*name*]#pause** *duration*	Defines a pause of *duration* milliseconds
4	***node*(pf-callp)[*name*]#...**	Repeat step 2 and/or step 3 to define a tone sequence. Always when you enter a **play** or **pause** command, it is appended to the already existing tone.
5	***node*(pf-callp)[*name*]#flush-play-list**	Resets the tone cadence. Same as deleting and re-creating the tone.

Example: Define the Belgian special information tone

The first line defines the first element of the tone: 330ms of 950Hz at –4dB. The second line the element that is played when the first element has finished: 330ms of 144Hz at –4dB, and so on. The last line defines a pause of 1 second after the three tones. The cadence is repeated infinitely.

```
node(cfg)#profile call-progress-tone belgianSpec
node(pf-callp)[belgian~]#play 330 950 -4
node(pf-callp)[belgian~]#play 330 1400 -4
node(pf-callp)[belgian~]#play 330 1800 -4
node(pf-callp)[belgian~]#pause 1000
```

Tones and pauses can be arbitrarily sequenced up to a number of 10 elements per call-progress-tone. The default call-progress-tone is an empty tone. The total number of different *play* elements across all configured call-progress-tones must not exceed 15 (an error is thrown if it does). If the call-progress-tone consists of only one element, this element has infinite duration. The *duration* parameter is ignored in this case.

Configure tone-set profiles

A tone-set profile maps one call-progress-tone profile to each internal call-progress-tone. A tone-set profile typically includes all the call-progress-tones for one country.

Procedure: To configure a tone-set profile

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#profile tone-set** *name*	Creates tone-set *name* and enters tone-set profile configuration mode.

Step	Command	Purpose
2	***node*(pf-tones)[*name*]#map call_progress_tone { busy-tone | confirmation-tone| congestion-tone | dial-tone | hold-tone | release-tone | ringback-tone | special-dial-tone | special-information-tone | waiting-tone } *call-progress-tone***	Map a call-progress-tone profile to an internal tone. An internal tone represents the call event for which a tone indication can be provided. Use the CLI help to get a list of all available events.
3		Repeat step 2 for all internal tone events.
4	[*name*](**pf-tones**)[*name*]#**[dtmf-signal-level**	Defines the output level of DTMF signals generated lcoally. This applies also to relayed DTMF signals.

Example: Configuring a tone-set

The following example shows how to configure a tone-set profile for UK.

```
node(cfg)#profile tone-set UK
node(pf-tones)[UK]#map call_progress_tone dialtone dialUK
node(pf-tones)[UK]#map call_progress_tone alertingtone ringUK
node(pf-tones)[UK]#map call_progress_tone busytone busyUK
```

Enable tone-set profile

A call on the SmartNode always has two signaling protocol endpoints. At the moment it is only possible to play locally generated tones on PSTN endpoints (ISDN, FXS) and not on IP based signaling endpoints (H.323, SIP). Dependent on the configuration several combinations of signaling protocol endpoints are possible (ISDN–ISDN, H.323-ISDN, FXS-SIP etc.). The SmartNode will always generate the tones locally and play it on the PSTN line as long as the other endpoint doesn't notifies availability of inband information and the PSTN endpoint is NOT of type ISDN-USER or FXO. If availability of inband information will be notified by one endpoint, the bearer channel already contains the necessary tone information and must not be generated locally.

If the user has not specified a tone-set profile, the default tone-set will be taken to generate the local inband information. For enabling a user defined tone-set profile on a specific interface proceed as follows.

Procedure: To assign a tone-set profile to a PSTN interface

Mode: Interface

Step	Command	Purpose
1	***node*(ctx-cs)[switch]#interface** *if-type if-name*	Enter interface configuration mode.
2	***node*(if-*type*)[*if-name*]#use profile tone-set** *name*	Assign a user defined tone-set profile to an interface.

Example: Assign tone-set profiles to an ISDN interface

The example shows how to use the SWISS tone-set for the CS context, and use the USA tone-set for an individual interface.

```
node(cfg)#context cs
node(ctx-cs)[switch]#interface isdn bri0
node(if-isdn)[bri0]#use profile tone-set USA
```

Show call-progress-tone and tone-set profiles

Use the show commands to display the call-progress-tone profiles as well as the tone-set profiles.

Procedure: To show call-progress-tone profiles

Mode: Administrator execution

Step	Command	Purpose
1	***node*#show profile call-progress-tone** [*name*]	Display all call-progress-tone profiles or a specific with name *name*

Example: Show call-progress-tone profile

The following example shows how to display the call-progress-tone profiles.

```
node#show profile call-progress-tone belgianSpec
Profiles:
---------

belgianSpec:
  Play  330ms (950Hz at -4dB)
  Play  330ms (1400Hz at -4dB)
  Play  330ms (1800Hz at -4dB)
  Pause 100ms
```

Procedure: To show tone-set profiles

Mode: Administrator execution

Step	Command	Purpose
1	***node*#show profile tone-set** [*name*]	Display all tone-set profiles or a specific with name *name*

Example: Show tone-set profile

The following example shows how to display the tone-set profile.

```
node#show profile tone-set test
Tone Profile: test
==================

 Used:                                  by 0 module(s)
 DTMF Duration:                         80ms
 DTMF Interspace:                       80ms

 Tones
 -----

  dial-tone:                            belgianSpec
  ringback-tone:                        defaultAlertingtone
  hold-tone:                            defaultHoldtone
  waiting-tone:                         defaultWaitingtone
  confirmation-tone:                    defaultConftone
  busy-tone:                            defaultBusytone
  congestion-tone:                      defaultCongestiontone
  release-tone:                         defaultReleasetone
  special-information-tone:             defaultSItone
  special-dial-tone:                    defaultSDtone
```

Example: The following example shows how to configure a tone-set profile for UK and apply it to the isdn interface bri0.

Create the call-progress-tone profiles:

```
node(cfg)#profile call-progress-tone dial-UK
node(pf-callp)[dial-UK]#play 5000 350 0 440 0

node(pf-callp)[dial-UK]#profile call-progress-tone alerting-UK
node(pf-callp)[alertin~]#play 400 400 0 450 0
node(pf-callp)[alertin~]#pause 200
node(pf-callp)[alertin~]#play 400 400 0 450 0
node(pf-callp)[alertin~]#pause 2000

node(pf-callp)[alertin~]#profile call-progress-tone busy-UK
node(pf-callp)[busy-UK]#play 400 400 0
node(pf-callp)[busy-UK]#pause 400
node(pf-callp)[busy-UK]#exit


Create the tone-set profile:

node(cfg)#profile tone-set UK
node(pf-tones)[UK]#map call_progress_tone dialtone dial-UK
node(pf-tones)[UK]#map call_progress_tone alertingtone alerting-UK
node(pf-tones)[UK]#map call_progress_tone busytone busy-UK
node(pf-tones)[UK]#exit


Assign the tone-set to the isdn interface bri0

node(cfg)#context cs
```

```
node(ctx-cs)[switch]#interface isdn bri0
node(if-isdn)[bri0]#use profile tone-set UK
```

Chapter 43 FXS port configuration

Chapter contents

Introduction 538
Shutdown and enable FXS ports 538
Bind FXS ports to higher layer applications 539
Configure country-specific FXS port parameters 539
Other FXS port parameters 540
Example 541

Introduction

This chapter provides an overview of POTS signaling and SmartNode FXS ports and describes the tasks involved in configuring FXS ports in SmartWare.

This chapter includes the following sections:

- Shutdown and enable FXS ports (see page 538)
- Bind FXS ports to higher layer protocols (see page 540)
- Configure country specific FXS port parameters (see page 539)
- Configure other FXS port parameters (see page 540)
- Select a low-bit-rate codec for FXS ports
- Example

FXS stands for *foreign exchange station* and is the exchange side of a POTS (plain old telephone system) line. Even though POTS is seen as old technology it still plays an important part in today's telecommunications networks. There are still a large number of analog phone sets in use worldwide and will do so in the future. These analog devices, be they phones, facsimile machines and the like, represent a large investment and it is desirable to have the technical means to hook such devices to a Voice over IP enabled network. POTS signaling.

The signaling procedure used on FXS ports is different throughout different countries, but the basic idea is to use the POTS line itself as a current loop which signals off-hook and on-hook of the handset. Going off-hook establishes a connection between the phone and whatever is on the other side (CO switch, SmartNode, etc.).

Shutdown and enable FXS ports

FXS ports are enabled by default. They also can be configured when they are active. But keep in mind that configuration of an active port temporarily disables the port for a short time (some milliseconds).

Mode: Context CS

Step	Command	Purpose
1	**(config)#port fxs** *slot port*	Enter FXS port configuration mode
2	**(prt-fxs)[***slot/port***]#shutdown**	Shutdown the port. All active calls are dropped!
3	**(prt-fxs) [***slot/port***]#no shutdown**	Activate the port.

Bind FXS ports to higher layer applications

An FXS port needs to be associated to an fxs interface in a CS context. The same mechanism of *encapsulation* and *binding* is used as known for e.g. Ethernet ports (see "Interfaces, Ports, and Bindings" on page 47).

Procedure: To bind an FXS port to an FXS interface of a CS context

Mode: Port FXS

Step	Command	Purpose
1	***node*(config)#port fxs** *slot port*	Enter configuration mode for FXS port
2	***node*(prt-fxs)[*slot/port*]#encapsulation cc-fxs**	Sets the encapsulation for the port. cc-fxs designs the encapsulation is a context CS interface
3	***node*(prt-fxs)[*slot/port*]#bind interface** *interface*	Binds the port to an interface in a CS context

Configure country-specific FXS port parameters

Unlike ISDN, POTS is heavily country specific even though there is a good chance that a phone for one country works reasonably good in another country. Country specific settings are contained in a so-called *fxs profile* which is integrated in the firmware of the SmartNode. It can be updated independently from the firmware by means of tftp download.

As there are more than 190 countries, SmartWare does not support all country parameters, so make sure that the country parameter for your country is available as a profile, before you begin operation.

Procedure: Configure country-specific FXS parameters

Mode: Port FXS

Step	Command	Purpose
1	***node*(config)#port fxs** *slot port*	Enter configuration mode for FXS port
2	***node*(prt-fxs)[*slot/port*]#use profile fxs** *profile*	Select a profile containing the country specific settings of the port attributes (ring voltage etc.). The available country profiles are listed when entering this command. The names listed are composed as follows: **ISO3166-1-Alpha-2 2 digit country code** Examples (currently available profiles): • *ch* Switzerland • *etsi* ETSI EG 201 188 Configuration (Europe) • *gb* Great Britain • *nl* Netherlands • *us* United States of America/Canada • *us-115vpp* USA, higher ring voltatge (115 Vpp) • *za* South Africa The default profile (not displayed in the list when entering the command) is *etsi*.

Other FXS port parameters

This section describes the commands available for the configuration of an FXS port.

Procedure: Configure the FXS port parameters

Mode: Configure

Step	Command	Purpose
1	***node*(config)#port fxs** *slot port*	Enter FXS port configuration mode
2 optional	*node*(prt-fxs) [*slot/port*]#[no] battery-reversal	Reverses the line polarity at connect/disconnect of the call. This might be required by certain PBX to work correctly. Default: disabled.
3 optional	*node*(prt-fxs) [*slot/port*]#end-of-call-signaling { busy-tone | { loop-break <duration> } }	Selects the method how SmartNode signals the end of a call to the connected analog terminal, playing a busy-tone or interrupting the loop-current for a certain time (duration in ms). Default: busy-tone.
4 optional	*node*(prt-fxs)[*slot/port*]#caller-id format { bell | etsi }	Specifies which line protocol is used for caller-id transmission. Use bell for US/Canada, etsi for Europe. Caller-id is enabled/disabled in the higher layer application (interface fxs in context CS). Default: etsi
5 optional	*node*(prt-fxs)[*slot/port*]#[no] caller-id attenuation *attenuation*	Attenuates the modulated caller-id signal (dB). Default: disabled.
6 optional	*node*(prt-fxs) [*slot/port*]#flash-hook-duration *duration*	Specifies for what time the connected device goes on-hook for flash-hook signaling (ms). When entering this command without an argument, the default duration of 1000ms is applied. Default: 1000ms. In US or Canada, try 350ms. To ensure that the flash-hook event can be relayed over SIP or H.323 in analog line extension applications, disable all local call features in the fxs interface in the cs context: no call-waiting, no additional-call-offering, no call-hold.
7 optional	*node*(prt-fxs) [*slot/port*]#[no] pulse-dialing	Enables the port for use with pulse dialing terminals. Recommended only when a terminal is connected that allows pulse dialing only. Enabling pulse dialing disables the use of all flash-hook features. Default: Disabled.

Mode: IC voice in system

Example

The following example shows how to enter the configuration mode for FXS port 0/0, configure it with typical US settings, and bind it to an interface named *fxs00* in context CS *switch*.

```
172.16.40.71>enable
172.16.40.71#configure
172.16.40.71(cfg)#port fxs 0 0
172.16.40.71(prt-fxs)[0/0]#use profile fxs us
172.16.40.71(prt-fxs)[0/0]#caller-id format bell
172.16.40.71(prt-fxs)[0/0]#flash-hook-duration 350
172.16.40.71(prt-fxs)[0/0]#encapsulation cc-fxs
172.16.40.71(prt-fxs)[0/0]#bind interface fxs00 switch
172.16.40.71(prt-fxs)[0/0]#exit
172.16.40.71(cfg)#system
```

Chapter 44 FXO port configuration

Chapter contents

Introduction 543
Shutdown and enable FXO ports 543
Bind FXO ports to higher layer applications 543
Configure country specific FXO port parameters 544
Other FXO port parameters 544

Introduction

This chapter provides an overview of POTS signaling and SmartNode FXO ports and describes the tasks involved in configuring FXO ports in SmartWare.

This chapter includes the following sections:

- Shutdown and enable FXO ports (see page 543)
- Bind FXO ports to higher layer protocols (see page 544)
- Configure country specific FXO port parameters (see page 544)
- Configure other FXO port parameters (see page 544)

FXO stands for *foreign exchange office* and is the subscriber side of a POTS (plain old telephone system) line. An FXO port on the SmartNode simulates thus an analog phone set, which must actively go on-hook and off-hook, detect ringing and caller-id, and dial the called-party number using DTMF keypad.

Shutdown and enable FXO ports

FXO ports are enabled by default. They also can be configured when they are active. But keep in mind that configuration of an active port temporarily disables the port for a short time (some milliseconds).

Mode: Configure

Step	Command	Purpose
1	**(config)#port fxo** *slot port*	Enter FXO port configuration mode
2	**(prt-fxo)[***slot/port***]#shut-down**	Shutdown the port. All active calls are dropped!
3	**(prt-fxo) [***slot/port***]#no shut-down**	Activate the port.

Bind FXO ports to higher layer applications

An FXO port needs to be associated to an fxo interface in a CS context. The same mechanism of *encapsulation* and *binding* is used as known for e.g. ethernet ports (see section "Interfaces, Ports, and Bindings" on page 47).

Procedure: To bind an FXO port to an fxo interface of a CS context

Mode: Port FXO

Step	Command	Purpose
1	***node*(config)#port fxo** *slot port*	Enter configuration mode for FXO port
2	***node*(prt-fxo)[*slot/port*]#encapsulation cc-fxo**	Sets the encapsulation for the port. cc-fxo designs the encapsulation is a context CS interface
3	***node*(prt-fxo)[*slot/port*]#bind interface** *interface*	Binds the port to an interface in a CS context

Configure country specific FXO port parameters

Unlike ISDN, POTS is heavily country specific even though there is a good chance that a setting for one country works reasonably good in another country. Country specific settings are contained in a so-called *fxo profile* which is integrated in the firmware of the SmartNode. It can be updated independently from the firmware by means of tftp download.

As there are over 190 countries SmartWare of course does not support all different country parameters. Thus before operation make sure that the country parameter for your country is available as profile.

Procedure: Configure country specific FXO parameters

Mode: Port FXO

Step	Command	Purpose
1	***node*(config)#port fxo** *slot port*	Enter configuration mode for FXO port
2	***node*(prt-fxo)[*slot/port*]#use profile fxo** *profile*	Select a profile containing the country specific settings of the port. Available profiles are: • 'etsi' according to ETSI standard • 'fcc68_25Hz' for the United States of America / Canada, according to the FCC86 standard, with 25Hz ringing detection. • 'fcc68_50Hz' for the United States of America / Canada, according to the FCC86 standard, with 50Hz ringing detection. • 'tbr21_25Hz', according to the TBR 21 standard, with 25Hz ringing detection. • 'tbr21_50Hz', according to the TBR 21 standard, with 50Hz ringing detection. The default profile (not displayed in the list when entering the command) is etsi.

Other FXO port parameters

This section describes the commands available for the configuration of an FXO port.

Procedure: Configure the FXO port parameters

Mode: Configure

Step	Command	Purpose
1	***node*(config)#port fxo** *slot port*	Enter FXO port configuration mode
2 optional	*node*(prt-fxo)[*slot/port*]#caller-id format { bell | etsi }	Specifies which line protocol is used for caller-id transmission. Use bell for US / Canada, etsi for Europe. If caller-id is not enabled or wrong configured, detection of caller-id is not possible. Default: etsi
3 optional	*node*(prt-fxo) [*slot/port*]#flash-hook duration *duration*	Specifies for what time the SmartNode should go on-hook to signal flash-hook to the CO. Default of *duration* is 200ms, for US and Canada try 350ms or higher.

Chapter 45 H.323 gateway configuration

Chapter contents

\Introduction 547
Gateway configuration task list 548
Binding the gateway to an IP interface 548
Enable the gateway 548
Configure registration authentication service (RAS) (Optional) 549
Configure H.235 Security (optional) 550
H.235 configuration 551
Advanced configuration options (optional) 554
Enabling H.245 Tunneling 554
Enabling the fastconnect procedure 555
Enabling the early H.245 procedure 555
Changing the TCP port for inbound call-signaling connections 556
Configuring the traffic class for H.323 signaling 556
Setting the response timeout 556
Setting the connect timeout 557
Configuring the terminal type for registration with the gatekeeper 557
Troubleshooting 558

Introduction

This chapter provides an overview of the H.323 gateway and describes the tasks involved in its configuration.

A gateway is always needed when communication is required between different networks. A gateway provides:

- Data format translation, e.g. audio and video CODEC translation
- Control signaling translation, e.g. call setup and tear-down functionality on both sides of a network.

The gateway manages connections between two different contexts, essentially functioning as a protocol converter. Additionally it also contains general gateway configuration parameters.

One type of gateway is the H.323 gateway and is an implementation of the ITU-T H.323 Version 4 standard. The H.323 gateway has bindings to interfaces on the two different contexts. The CS Context has H.323 CS interfaces and the IP context has IP interfaces. The H.323 CS interfaces are explained in detail in chapter 38, "H.323 interface configuration" on page 431.

The H.323 interfaces in the CS context must be explicitly bound to a H.323 gateway instance, and the H.323 gateway must be bound explicitly to an IP interface in the IP context. Figure 77 illustrates the relationship of the H.323 gateway to the contexts and their interfaces.

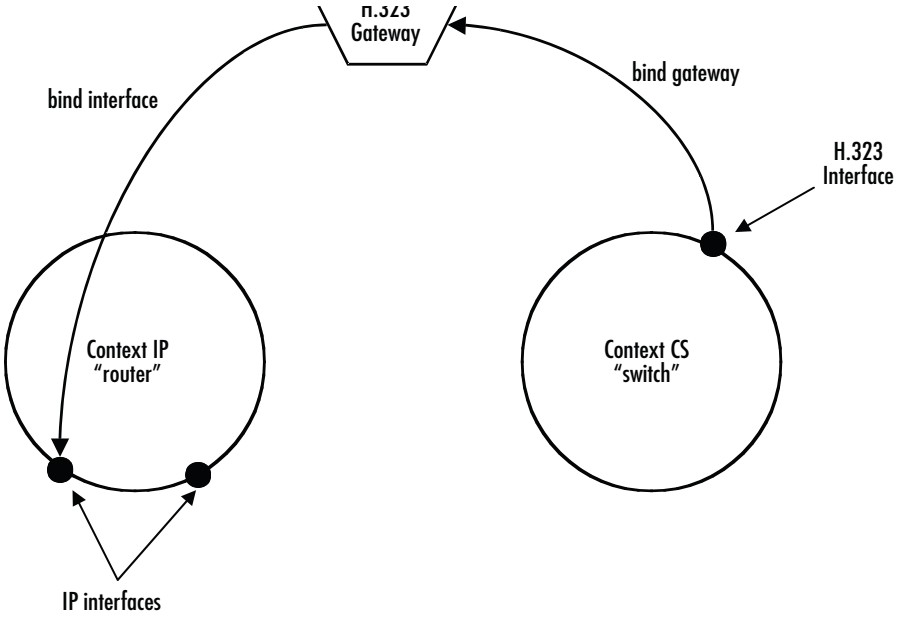


Figure 77. Gateway between IP and CS contexts

SmartWare currently supports only one instance of the H.323 gateway. The default name of the H.323 gateway is h323. The H.323 gateway is enabled by default.

Gateway configuration task list

This chapter describes the configuration of the H.323 gateway. Some parameters can be configured in the gateway configuration mode and may be overwritten in another configuration mode, For example, in the H.323 CS interface. For example, the default VoIP profile to be used with the gateway is configured in the gateway configuration mode. However it can be overwritten by another VoIP profile specified in the H.323 CS interface. All possible configurations, which are involved in a specific configuration topic are described in the respective configuration task.

- Configure datapath related settings
- Binding the gateway to an IP interface
- Enable the Gateway
- Configure Registration Authentication Service (RAS) (Optional)
- Configure H.235 Security (Optional)
- Advanced configuration options (Optional)

Binding the gateway to an IP interface

Binding the gateway to one of the available IP interfaces is required to allow the gateway to determine the local IP address it should use. The gateway needs to know the local IP address For example, when it needs to tell the remote gateway in the call-signaling to which IP address the remote gateway should send the media streams (RTP data). The gateway always uses the IP address of the interface to which it is bound, when a local IP address is required. The following procedure describes how to bind the gateway to a local IP interface.

Mode: Gateway H.323

Step	Command	Purpose
1	*node*(gw-h323)[h323]#**bind interface** *if-name*	Binds the gateway to the IP interface if-name.

Example: Binding the gateway

The following example shows how to bind the gateway to an IP interface.

```
node>enable
node#configure
node(cfg)#gateway h323 h323
node(gw-h323)[h323]#bind interface eth0
```

Enable the gateway

In order to become active the H.323 gateway must be enabled. The following procedure enables the H.323 gateway:

Mode: Gateway H.323

Step	Command	Purpose
1	*node*(gw-h323)[*h323*]#**no shutdown**	Enable the H.323 gateway

Example: Enabling an H.323 gateway

The following example shows how to enable an already defined H.323 Gateway.

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

Configure registration authentication service (RAS) (Optional)

The H.323 gateway can either work in combination with a gatekeeper, which uses the RAS protocol for communication with the gateways or it can be used for direct calls between gateways without a gatekeeper. If you do not use a gatekeeper, you can skip this section. Otherwise you need to provide some information for the registration with the gatekeeper using the procedures described in this chapter.

The SmartNode can register one or more names, E.164 numbers or gatekeeper prefixes, which can be specified using this procedure too. Furthermore the gatekeeper discovery method must be specified as automatic or manual.

Normally the remote IP address which could be specified in the CS interface is not set if gatekeepers are used, because the gatekeeper supplies the remote destination IP address.

Redundancy is of great importance in communication networks. An H.323 gatekeeper offers a critical service in a network – failure of a single gatekeeper may result in non-availability of the voice service in the entire network. The SmartNode allows configuring up to three different gatekeeper s, when using the manual gatekeeper discovery method. These gatekeepers are tried in a round-robin fashion. Once communication with one of the gatekeepers is lost the SmartNode falls back to the next gatekeeper in the list.

Procedure: To enable the registration authentication service (RAS)

Mode: Gateway H.323

Step	Command	Purpose
1	*node*(gw-h323)[h323]#**gatekeeper-discovery auto** *[gkid]* or *node*(gw-h323)[h323]#**gatekeeper-discovery manual** *ip-address ip-port [gkid]*	Specify that gatekeeper discovery has to be done automatically or Specify the gatekeeper explicitly. You can repeat this command to add multiple gatekeepers.
2	*node*(gw-h323)[h323]#**alias h323-id** *name* or *node*(gw-h323)[h323]#**alias e164** *number*	Add an H.323_ID or E.164 alias for registration.
3		Repeat step 2 to add more than one alias to the configuration.
4	*node*(gw-h323)[h323]#**supported-prefix** *prefix*	Add a gatekeeper prefix to be registered on the gatekeeper
5		Repeat step 4 to add more than one prefix to the configuration.
6	*node*(gw-h323)[h323]#**ras**	Enable the RAS protocol to make use of gatekeepers and initiate gateway registration.

Example: Configuring RAS

The following example shows how to configure the registration authentication service (RAS) for a SmartNode in an H.323 network.

```
node>enable
node#configure
node(cfg)#gateway h323 h323
node(gw-h323)[h323]#gatekeeper-discovery auto
node(gw-h323)[h323]#alias h323-id Berne1
node(gw-h323)[h323]#alias e164 007
node(gw-h323)[h323]#alias e164 *5
node(gw-h323)[h323]#alias e164 19421
node(gw-h323)[h323]#supported-prefix 03198525
node(gw-h323)[h323]#ras
```

Configure H.235 Security (optional)

H.235 is an ITU-T Recommendation for security and encryption for H-series (H.323 and other H.245-based) multimedia terminals. It describes enhancements within the framework of the H.3xx-Series Recommendations to incorporate security services such as Authentication and Privacy (data encryption).

H.235v2 Annex D provides H.323 RAS and H.225 message authentication and integrity check thus thwarting any replay and spoofing attacks on H.323 calls. If H.235 is switched on, the following security attacks are thwarted:

- Denial of Service attacks
- Man-in-the-middle attacks
- Replay attacks (replay of recorded messages)
- Spoofing
- Connection hijacking

Among other information such as time stamp, sender and general ID, the H.235 needs a password for crypto token generation. Since this password is intelligible when being configured by means of a Telnet session or displayed in a running configuration, it is possible to configure an encrypted password, which will be decrypted on the SmartNode. For decryption a master password is needed. Configuration of the master password should not be done over insecure links (links subject to wire-tapping). It is recommended to do so in a secure network (local area network) only (before delivery to the customer).

Henceforth, the H.235 password can be reconfigured securely even over insecure links.

To generate an H.235 encrypted password by means of the master password as key, the password encryption tool is used ('getcryptopassword.exe'). The usage of the Windows based command line tool is as follows:

```
getcryptopassword <h235-password> <master-password>
```

The H.235 password must be a random alphanumeric character string of 1 through 12 characters (e.g. 12ygR34230kG). The master password must be a 32 digit hex number (characters 0-9, a-f). To achieve best encryption security, choose a random value (no repeating character sequences). The tool generates the encrypted H.235 password and the hash of the master password. The encrypted H.235 password is then to be used for remote (over insecure link) configuration of the H.235 password. The hash value of the master password can be used to verify proper configuration of all parameters. The command **show h235security** displays all H.235 settings including a hash value of the master password. If this value is identical to the hash value output by the tool *gencryptopassword.exe*, the configuration of the master password was successful. Note that this last verification step can be done securely even over insecure links (subject to wire-tapping) since the algorithm used for hash value calculation is a mathematical one-way function (virtually impossible to derive the password from the hash value). To enable H.235 security on H.323 perform the steps described below.

Procedure: To enable H.235 security on H.323 gateway

H.235 configuration

You can control on a per-message-type basis which RAS messages are sent H.235 signed and of which RAS messages the H.235 signature shall be verified. Therefore the commands **h235-security ras-auth-int-rx** and **h235-security ras-auth-int-tx** have a new optional parameter that specifies the message type. The new format is:

- [no] h235-security ras-auth-int-rx [<msg>]
- [no] h235-security ras-auth-int-tx [<msg>]

Mode: Gateway H.323

Step	Command	Purpose
1	*node*(gw-h323)[h323]#**h235security master-password** *master-password*	Sets the master password (32 hex digits, 0–9, A–F) with which the H.235 password is decrypted. **Note** Configure the master password only over secure links (e.g. in LAN environments only or with serial connection), which cannot be wiretapped.
2	**C:\getcryptopassword** *h235-password master-password*	Generates H.235 password by means of the master password with the encryption tool.
3	*node*(gw-h323)[h323]#**h235security password** *h235-password* **encrypted** or *node*(gw-h323)[h323]#**h235security password** *h235-password*	Sets the password used for crypto token calculation. The password is entered encrypted. The password to be entered is the output of the tool *getcryptopassword.exe*. Configures the password used for crypto hashed token calculation. The password is entered in clear text (min. 1, max. 12 alphanumeric characters). **Note** Do not use this command over insecure links (subject to wiretapping). If you enter the password as clear text, you don't need to configure a master-password.
4	*node*(gw-h323)[h323]#**h235security time-window** *time-window*	Sets the time window used for timestamp comparison by H.235. If a received H.323 message with H.235 crypto token has a timestamp outside the time window (relative to the local time) the message is refused.
5	*node*(gw-h323)[h323]#**h235security version {v1 | v2}**	There are two H.235 versions, use *v1* if *v2* does not work. In v1, *sender-id* and *general-id* must not be specified.
6	*node*(gw-h323)[h323]#**h235-security ras-auth-int-rx [**<msg>**]** *or* *node*(gw-h323)[h323]#**no h235-security ras-auth-int-rx [**<msg>**]**	Enables or disables H.235 security for received RAS packets. *msg* is the message type.

Step	Command	Purpose
7	*node*(gw-h323)**[**h323**]#h235-security ras-auth-int-tx [**<msg>**]** *or* *node*(gw-h323)**[**h323**]#no h235-security ras-auth-int-tx [**<msg>**]**	Enables or disables H.235 security for transmitted RAS packets. *msg* is the message type.
8	*node*(gw-h323)[h323]#**h235-security q931-auth-int** *or* *node*(gw-h323)[h323]#**no h235-security q931-auth-int**	Enables or disables H.235 security for call-signaling.
9	*node*(gw-h323)[h323]#**h235security**	If all parameters are set, enables H.235 security. Otherwise returns an error message.
10	*node*(gw-h323)[h323]#**show h235security**	Shows status of H.235 security module.
11	*node*(gw-h323)[h323]#**debug h235security [detail** *debug-level***]**	Enables the H.235 debug monitor to display information for every received and sent H.323 signaling message.

For example to disable signature verification of RCF messages enter the command **no h235-security ras-auth-int-rx rcf**. When not specifying the message type, you change the behavior of all message types. Additionally there are special message-types that manipulate the behavior of message-groups:

```
'request' ==> ARQ, BRQ, DRQ, GRQ, IRQ, LRQ, RAI, RRQ, SCI, URQ
'response' ==> ACF, BCF, DCF, GCF, IACK, IRR, LCF, RAC, RCF, RIP, SCR, UCF
'reject' ==> ARJ, BRJ, DRJ, GRJ, INAK, LRJ, RRJ, URJ
E.g. 'no h235-security ras-auth-int-rx reject' disables verification of the H.235
signature of all received reject messages, e.g. GRJ, RRJ, ARJ.
```

The default setting is:

```
h235-security ras-auth-int-rx request
h235-security ras-auth-int-rx response
no h235-security ras-auth-int-rx reject
h235-security ras-auth-int-tx request
h235-security ras-auth-int-tx response
no h235-security ras-auth-int-tx reject
```

The command **show h235-security** shows the current setting.

Example: Switch on H.235 security

The following example shows how to use the password encryption tool and how to enable H.235 security. Additionally the H.235 security debug monitor is enabled.

```
Generate the encrypted H.235 password from 'myh235pwd':

C:\>getcryptopassword myh235pwd 12d3f4e76a83c6dd1067a6d34fe5cb21

H.235 Password          : myh235pwd
Encrypted H.235 Password: 21dafa5dfc7399e5cef9cc138dabd22f
Master Password         : 12d3f4e76a83c6dd1067a6d34fe5cb21
Hash of Master Password : bc210d2244a1afd2e7da7a54a1a8c179403220c4

C:\>

Configure and enable H.235:

172.16.224.102(cfg)#gateway h323 h323
172.16.224.102(gw-h323)[h323]#h235security master-password
12d3f4e76a83c6dd1067a6d34fe5cb21
172.16.224.102(gw-h323)[h323]#h235security password
21dafa5dfc7399e5cef9cc138dabd22f encrypted
172.16.224.102(gw-h323)[h323]#h235security time-window 100
172.16.224.102(gw-h323)[h323]#h235security version v2
172.16.224.102(gw-h323)[h323]#h235security ras-auth-int-tx
172.16.224.102(gw-h323)[h323]#h235security ras-auth-int-rx
172.16.224.102(gw-h323)[h323]#h235security q931-auth-int
172.16.224.102(gw-h323)[h323]#show h235security

H.235 SECURITY SETTINGS
-----------------------
  H.235 Security              : Disabled
  H.235 Module Version        : 2.02.01
  H.235 Version               : 2
  Sender ID                   : NODE13
  General ID                  : GK01
  Time Window                 : 100 seconds
  Master Password Hash        : bc210d2244a1afd2e7da7a54a1a8c179403220c4
  Debug Monitor:              : Disabled

172.16.224.102(gw-h323)[h323]#
172.16.224.102(gw-h323)[h323]#debug h235security detail 3
172.16.224.102(gw-h323)[h323]#h235security
172.16.224.102(gw-h323)[h323]#14:27:35  H235  > Info: H.235 was started successfully
```

Advanced configuration options (optional)

Enabling H.245 Tunneling

If H.245 Tunneling is enabled, H.245 messages use the same TCP connection as the H.323 signaling. H.245 Tunneling is disabled by default. If enabled, it only takes place when both parties agree. When experiencing problems with establishing H.323 calls, disabling H.245-Tunneling may help.

Mode: Gateway H.323

Step	Command	Purpose
1	*node*(gw-h323)[h323]#**h245-tunneling**	Enables H.245 tunneling.

Example: Enabling H.245 tunneling

The following example shows how to enable H.245 tunneling on an already defined H.323 Gateway.

```
node(cfg)#gateway h323 h323
node(gw-h323)[h323]#h245-tunneling]
```

Enabling the fastconnect procedure

If the fastconnect procedure is enabled, no separate H.245 connection is opened and all media channel specific messages are carried within the H.225 call signaling connection. Fastconnect is disabled by default. If enabled, it only takes place when both parties agree. Fastconnect can be enabled using the following procedure:

Mode: Gateway H.323

Step	Command	Purpose
1	*node*(gw-h323)[h323]#**faststart**	Enables the fastconnect procedure.

Example: Enabling fastconnect

The following example shows how to enable the fastconnect procedure on an already defined H.323 Gateway.

```
node(cfg)#gateway h323 h323
node(gw-h323)[h323]#faststart
```

Enabling the early H.245 procedure

If the early H.245 procedure is enabled, the H.245 connection is opened as soon as possible instead of waiting for the call signaling connect message. Early H.245 is disabled by default. If enabled, it only takes place when both parties agree. The early H.245 procedure can be enabled using the following procedure:

Mode: Gateway H.323

Step	Command	Purpose
1	*node*(gw-h323)[h323]#**early-h245**	Enables the early H.245 procedure.

Example: Enabling early H.245

The following example shows how to enable early H.245 on an already defined H.323 Gateway.

```
node(cfg)#gateway h323 h323
node(gw-h323)[h323]#early-h245
```

Changing the TCP port for inbound call-signaling connections

The default TCP listening port for inbound call-signaling connections is per default 1720 as defined in the H.323 standard. The following procedure describes how to change the port number.

Mode: Gateway H.323

Step	Command	Purpose
1	*node*(gw-h323)[h323]#**call-signaling-port** *port*	Define the TCP port to use for inbound call-signaling connections

Example: Defining an alternate call-signaling port

The following example shows how to define an alternate call-signaling port on an already defined H.323 Gateway.

```
node(cfg)#gateway h323 h323
node(gw-h323)[h323]#call-signaling-port 1721
```

Configuring the traffic class for H.323 signaling

The traffic class for H.323 signaling is configurable. The configured traffic class is used as additional routing criterion in the IP routing table.

Mode: Gateway H.323

Step	Command	Purpose
1	[*name*] **(gw-h323)**[*gateway*]**# call-signaling-trafficclass <traffic-class>**	Sets traffic class for H.323 signaling packets. The traffic class may be new or may already exist.

Setting the response timeout

Per default the H.323 gateway waits for 12s from the time it initiated a call towards the IP network until it terminates the call, if no response has been received. This time can be changed using the following procedure:

Mode: Gateway H.323

Step	Command	Purpose
1	*node*(gw-h323)[h323]# **timeout response** *seconds*	Defines the response timeout in seconds.

Example: Defining an alternate response timeout

The following example shows how to define an alternate response timeout of 6 seconds on an already defined H.323 Gateway.

```
node(cfg)#gateway h323 h323
node(gw-h323)[h323]#timeout response 6
```

Setting the connect timeout

Per default the H.323 gateway waits for 60s when the call is in the alerting phase for the call to be answered. If the call is not answered within that time, the call is dropped. The value of this timer can be changed using the following procedure:

Mode: Gateway H.323

Step	Command	Purpose
1	*node*(gw-h323)[h323]# **timeout connect** *seconds*	Defines the connect timeout in seconds.

Example: Defining an alternate connect timeout

The following example shows how to define an alternate connect timeout of 20 seconds on an already defined H.323 Gateway.

```
node(cfg)#gateway h323 h323
node(gw-h323)[h323]#timeout connect 20
```

Configuring the terminal type for registration with the gatekeeper

H.323 gatekeepers may differentiate between two terminal types (terminals and gateways) of the registrant. In some applications it is necessary for the gateway to register as a terminal, while in other applications it is necessary to register as a gateway.with the gatekeeper. The default terminal type is gateway. It can be changed using the following procedure: Usually you do not need to change this setting.

Procedure: Configure the registration type of the registration with the gatekeeper

Mode: Gateway H.323

Step	Command	Purpose
1	*node*(gw-h323)[h323]#**terminal-type { terminal | gateway }**	Set the registration type of the gatekeeper registration.

Example: Configuring RAS registration type

The following example shows how to configure the RAS registration type.

```
node(cfg)#gateway h323 h323
node(gw-h323)[h323]#terminal-type gateway
```

Troubleshooting

You can display basic status information of the H.323 gateway using the following command:

Mode: Configure

Step	Command	Purpose
1	*node*(gw-h323)[h323]# **show gateway h323 status [detail *level*]**	Displays H.323 gateway status information. The detail *level* parameter is a number in the range 0 to 5 and indicates how much detail should be displayed.

Example: Display H.323 gateway status information

The following example shows how to display H.323 gateway status information and a sample output of the command.

```
node(cfg)#show gateway h323 status
           H.323 Gateway:                          h323
           State:                                  UP
           Stack Handle:                           0xb6a70c
           RAS Engine
           State:                                UNREGISTERED
           Allocated Endpoints:                    0
           Allocated RAS Engines:                  1
           Allocated Control Channels:             0
           Allocated Outgoing Logical Slowstart Channels: 0
           Allocated Outgoing Logical Faststart Channels:0
           Allocated Incoming Logical Channels:0
```

The H.323 gateway also provides several debugging monitors to observe its dynamic behavior. These monitors allow efficient troubleshooting of H.323 problems. The most often used monitors are listed in the following table.

Command to enable the monitor	Output of the monitor
*node(cfg)#***debug gateway h323 error**	Logs all errors detected within the H.323 gateway
*node(cfg)#***debug gateway h323 tpktchan**	Logs all H.225 call signaling messages sent or received over the IP network.
*node(cfg)#***debug gateway h323 udpchan**	Logs all H.225 RAS messages sent or received over the IP network
*node(cfg)#***debug gateway h323 datapath**	Logs information related to media channels
*node(cfg)#***debug gateway h323 signaling**	Logs call signaling related information

Chapter 46 Context SIP gateway overview

Chapter contents

Introduction 560
Context SIP Gateway configuration task list 561
Creating a context SIP gateway 561
Creating a transport interface 562
Configuring the IP binding 562
Configuring a priority 562
Configuring a spoofed contact address 563
Binding location services 563
Enabling/disabling the context SIP gateway 563
Troubleshooting 564
Show status information 564
Debug commands 564
Configuration Examples 565
Example 1 565
Example 2 565
Example 3 565
Applications 566
Outbound Authentication 566
Inbound Authentication 567
Outbound Registration 568
Inbound Registration 570
B2B User Agent with Registered Clients 571

Introduction

This chapter provides an overview of the context SIP gateway. The main purpose of a context SIP gateway is forwarding and reception of SIP packets according to a RFC 3261 User Agent. In SmartWare, the context SIP gateway represents the interface between the Call-Router (Context CS) and the IP Router (Context IP). It is responsible for dispatching incoming initial SIP requests to the right call control SIP interface and for the determination of the right IP interface for outgoing SIP messages. This is also called *SIP Transport Routing*. Figure 78 shows where the context SIP gateway is located in SmartWare and how it interacts with other components.

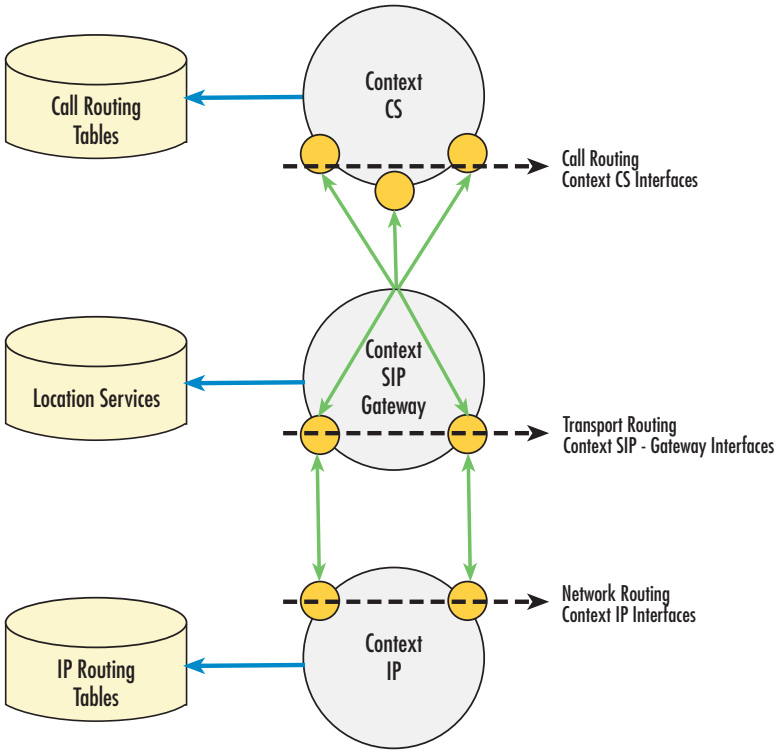


Figure 78. Routing Architecture

There is a relationship between the call control SIP interface and the context SIP gateway. Every call control SIP interface must be bound to a context SIP gateway. One of the responsibilities of this interface is to build Request-URI and to determine a possible outbound proxy. If a proxy is available, it represents the next SIP hop and all outgoing messages will be sent to this host. If no proxy has been configured, the messages will be sent to the Request-URI's host. Depending on these two SIP parameters, the context SIP gateway chooses the right outgoing IP interface. The IP address of the outgoing IP interface will be placed in all SIP and SDP headers that need a direct routable contact address (VIA, Contact).

In the other direction, the context SIP gateway dispatches incoming SIP requests according to the Request-URI and the From-URI. The call control SIP interface has configuration parameters called “remote” and “local” to build the Request-URI, the To-URI and the From-URI for outgoing requests. These parameters will also used for identifying the best matching call control SIP interface for incoming requests according to the following rule:

1. **From-URI-Host** *equal* **Remote**
   and
   Request-URI-Host *equal* **Local**
2. **From-URI-Host** *equal* **Remote**
3. **Request-URI-Host** *equal* **Local**
4. No match, the first configured will be taken

For detailed information about call control SIP interface configuration, see Chapter 39, "SIP interface configuration" on page 441.

Context SIP Gateway configuration task list

The following section describes how to create a new context SIP gateway and how to enter the configuration mode of an existing context SIP gateway. Additionally, it describes all commands and sub commands of the context SIP gateway configuration mode. All configuration tasks for a Context SIP Gateway are listed below.

- Create a context SIP gateway (see page 561)
- Create a transport interface (see page 562)
- Configure the IP binding (see page 562)
- Configure a priority (see page 562)
- Configure a spoofed contact address (see page 563)
- Bind location services (see page 563)
- Enable/Disable (see page 563)

Creating a context SIP gateway

The **context sip-gateway** command enters the configuration mode of a context SIP gateway. If the requested one does not exist, a new one will be created. The no form of the command removes an existing context SIP gateway. This command can be entered without specifying a name. In this case, the default name *sip* will be taken.

Mode: Configure

Step	Command	Purpose
1	[*node*]**(cfg)# [no] context sip-gateway** [*<name>*]	Creates/Destroys a context SIP gateway or enter configuration mode.

Creating a transport interface

The **interface** command enters the configuration mode of a transport interface. If the requested interface does not exist, a new one will be created. The no form of the command removes an existing transport interface.

Mode: Context SIP Gateway

Step	Command	Purpose
1	[*node*]**(sip-gw)[name]# [no] interface** *<name>*	Creates/Destroy sa transport interface or enter configuration mode.

Configuring the IP binding

The following commands specify the ip parameters for the transport layer. A listening socket for UDP and TCP will be created according to the entered parameters. If the ip-interface parameter exists, the ip address of the interface will be used. If the ip parameter does not exist, 0.0.0.0 will be used. If no port has been entered, the default SIP port 5060 will be used. The no form of the command removes an existing binding.

Mode: Transport Interface

Step	Command	Purpose
1	[*node*]**(sip-if)[name]# [no] bind** or [*node*]**(sip-if)[name]#bind interface** *ip-interface* or [*node*]**(sip-if)[name]#bind port** *<port>* or [*node*]**(sip-if)[name]#bind interface** *ip-interface* **port** *<port>*	Binds the transport interface to the IP layer or removes an existing binding.

Configuring a priority

The priority is used if a SIP request is sent out over an ip interface to which the context SIP gateway has no binding. In this case, the Contact (SIP/SDP) and VIA headers will get the ip address of the transport interface with the highest priority. The lower the priority number, then the higher the priority. The highest priority is 0. The no form of command sets the default priority to 0.

Mode: Transport Interface

Step	Command	Purpose
1	[*node*]**(sip-if)[name]# [no] priority** *<priority>*	Applies a priority to the transport interface.

Configuring a spoofed contact address

If the device is located behind a NAT, this command can be used to provide the Contact (SIP/SDP) and VIA headers of outgoing requests with the public ip address.

Mode: Transport Interface

Step	Command	Purpose
1	[*node*]**(sip-if)[name]# [no] spoofed-contact** *<address>*	Applies a spoofed contact address to the transport interface.

Binding location services

The **bind location-service** command binds predefined location services to the context SIP gateway. The no form of the command removes a bound location service. All bound location services define the domains and identities for the context SIP gateway. Identity features, such as outbound registration, inbound registration, and authentication, may or may not be executed, depending on the specific confirguation for the identity. Also, they provide transport properties, like Proxy or Traffic Class, and media configuration parameters, like VoIP Profile, SIP Profile or Tone Profile. For more information about configuration of location services and identities, see Chapter 51, "Location Service" on page 607.

Mode: Context SIP Gateway

Step	Command	Purpose
1	[*node*]**(sip-gw)[name]# [no] bind location-service** *locationservice*	Add a location service binding to this context SIP gateway or removes an existing one.

Enabling/disabling the context SIP gateway

The **shutdown** command disables the context SIP gateway. The **no shutdown** command enables the context SIP gateway.

Mode: Context SIP Gateway

Step	Command	Purpose
1	[*node*]**(sip-gw)[name]# [no] shutdown**	Disables the Context SIP Gateway. The no form of the command enables the Context SIP Gateway.

Troubleshooting

Show status information

Mode: Administrator Execution

Step	Command	Purpose
1	[*node*]**#show context sip-gateway** [*gw-name*] **[detail** *<level>*]	Displays status and configuration information about a context sip-gateway and its bound resources like the call-control sip interfaces.

Debug commands

Mode: Administrator Execution

Step	Command	Purpose
1	[*node*]**#debug context sip-gateway data-path [detail** *<level>*]	Logs information related to the media channels.
	[*node*]**#debug context sip-gateway error [detail** *<level>*]	Logs all errors detected within the Context SIP Gateway.
	[*node*]**#debug context sip-gateway registration [detail** *<level>*]	Logs information about user registration activities.
	[*node*]**#debug context sip-gateway signaling [detail** *<level>*]	Logs call signaling related information.
	[*node*]**#debug context sip-gateway transport [detail** *<level>*]	Logs all SIP messages sent or received over the IP network.

Configuration Examples

Example 1

This is the minimal configuration of a working context sip-gateway. It has one transport interface that is bound to the ip address 0.0.0.0 and to the port 5060. With this configuration, the context sip-gateway processes all SIP requests addressed to port 5060 independent on which ip interface they arrive.

```
context sip-gateway SIP-GW
  interface lan
    bind port 5060
context sip-gateway SIP-GW
  no shutdown
```

Example 2

The following example shows a context sip-gateway that is explicitly bound to a LAN-Side and a WAN-Side ip interface. In this case, only SIP requests sent to ip interface 'eth0' and 'pvc100' will be processed. This could be a Back-To-Back User Agent application that interconnects a private SIP environment and a public SIP environment.

```
context sip-gateway SIP-GW
  interface lan
    bind interface eth0 context router port 5060
  interface wan
    bind interface pvc100 context router port 5060
context sip-gateway SIP-GW
  no shutdown
```

Example 3

If special features like Outbound Registration, Inbound Registration or Authentication is required, one or more location services must be configured that enables theses capabilities in general, for a group of identities or just for a single identity. These location services must be bound to the context sip-gateway that is responsible to manage the location service's domains. For detailed description about location service configuration, see Chapter 50, "Authentication Service" on page 604.

```
context sip-gateway SIP-GW
  interface lan
    bind interface eth0 context router port 5060
  interface wan
    bind interface pvc100 context router port 5060
context sip-gateway SIP-GW
  bind location-service SIP_LS_1
  no shutdown
```

Applications

Outbound Authentication

The back-to-back user agent can provide credentials for authentication on another sip user agent or proxy. The username and password used for authentication must be configured in an authentication-service. If one or more realms are configured in the authentication-service, the credentials are only provided to challenges containing one of these realms. If no realm is configured, the credentials are provided to any realm.

In an authentication-service, there can be multiple usernames and passwords. An identity which should authenticate can direct the authentication outbound face to a pair of credentials. There can be multiple identities using the same credentials. An identity can also point to multiple credentials, but each of these credentials needs to be in another authentication-service with another realm. It is possible to authenticate to multiple realms with multiple credentials at the same time.

If the gateway has to provide credentials for unknown identities or for any identity which belongs to a certain domain, there can be a “default” identity-group configured. The authentication credentials configured in the identity-group “default” are used for any identity in this location-service that is not explicitly configured.

```
authentication-service AUTH_INALP
  realm inalp.com
  username hermes password Wh6Xbk9G= encrypted
  username john password Fa0Y9e4L= encrypted

authentication-service AUTH_ANY
  username bob password Co7s3bUp= encrypted

location-service INALP
  domain inalp.com
  domain patton.com

  identity-group default
    authentication outbound
      authenticate 1 authentication-service AUTH_ANY username bob

  identity 400
    authentication outbound
      authenticate 1 authentication-service AUTH_INALP username hermes
      authenticate 2 authentication-service AUTH_ANY username bob

  identity 500
    authentication outbound
      authenticate 1 authentication-service AUTH_INALP username hermes

  identity 600
    authentication outbound
      authenticate 1 authentication-service AUTH_INALP username john
```

If the gateway needs to provide authentication credentials on a sip request, the following procedure takes place:

1. Determine the location-service which provides credentials. The domain of the location service must match the host part of the from-uri and the location-service is bound to the context sip-gateway which sends the request.

2. Determine the identity which provides credentials. The name or the alias of the identity must match the user part of the from-uri. If there is no identity that matches and an identity-group with the name “default” is configured, the identity-group “default” is taken.
3. Determine the authentication-service which provides credentials. The authentication entries of the taken identity or identity-group are searched for an authentication-service that matches exactly the realm requested in the answer to our request. Then this authentication service is taken. If no match was found, an authentication service with no realm configured is taken.
4. Determine the authentication username which provides credentials. If the authentication entry of the identity which configures the taken authentication service has also configured a username this username is taken. If there is no username configured the name of the identity is taken as username.
5. Take the credentials in the authentication service with the according username and provide username and password for re-issuing the request.

If one of these steps has no result and fails, authentication is not possible for that request.

Inbound Authentication

The back-to-back user agent can challenge another sip user agent or proxy for authentication credentials. The username and password used for challenges must be configured in an authentication-service. There must be at least one realm configured in the authentication-service. The first realm configured is used for challenging requests.

In an authentication-service, there can be multiple usernames and passwords. An identity which should be challenged can direct the authentication inbound face to a pair of credentials. There can be multiple identities using exactly the same credentials. An identity can also point to multiple credentials, but only the first entry is used for challenging. If an identity points to multiple credentials, any of these credentials are accepted in the answer as long as it is valid for the challenged realm.

If the gateway has to challenge credentials for unknown identities or for any identity which belongs to a certain domain, there can be a “default” identity-group. The challenging credentials configured in the identity-group “default” are used for any identity in this location-service that is not explicitly configured.

```
authentication-service AUTH_PATTON
  realm patton.com
  username kevin password Wh6Xbk9G= encrypted
  username dirk password Fa0Y9e4L= encrypted
  username boss password Q9Gns6Nd4= encrypted

location-service PATTON
  domain patton.com

  identity-group default
    authentication inbound
      authenticate 1 authentication-service AUTH_PATTON username kevin

  identity 400
    authentication inbound
      authenticate 1 authentication-service AUTH_PATTON username kevin
      authenticate 2 authentication-service AUTH_PATTON username dirk

  identity 555
    authentication inbound
```

```
authenticate 1 authentication-service AUTH_PATTON username boss
```

If the gateway receives an incoming request without credentials, the following procedure takes place:

1. Determine the location-service which challenges credentials. The domain of the location service must match the host part of the request-uri and the location-service is bound to the context sip-gateway which handles the request.
2. Determine the identity which challenges credentials. The name or the alias of the identity must match the user part of the request-uri. If there is no identity that matches and an identity-group with the name “default” is configured, the identity-group “default” is taken.
3. Determine the authentication-service which challenges credentials. The first authentication entry of the taken identity or identity-group where a realm is configured is taken.
4. The first realm in the authentication-service is used for challenging the request. If no realm was found no challenging is possible.

If one of these steps has no result and fails, challenging is not possible for that request and the request is accepted.

If the gateway receives an incoming request with credentials, the following procedure takes place:

1. Determine the location-service which challenges credentials. The domain of the location service must match the host part of the request-uri and the location-service is bound to the context sip-gateway which handles the request.
2. Determine the identity which challenges credentials. The name or the alias of the identity must match the user part of the request-uri. If there is no identity that matches and an identity-group with the name “default” is configured, the identity-group “default” is taken.
3. Check credentials. All authentication entries of the taken identity or identity-group are compared with the provided credentials. If one matches the request is accepted.
4. An authentication entry matches if the username and password matches with the configured credentials and one realm in the authentication-service match exactly the realm challenged or the authentication-service has no realm configured.

If one of these steps has no result and fails, the request is treated as a request without credentials.

Outbound Registration

The back-to-back user agent can register identities on a sip registrar. Therefore, a registration outbound face with the register “auto” command is needed. The address of record is built with the domain of the location-service and the name of the identity. The REGISTER request is sent to the configured registrar or, if missing, to the domain configured in the location-service.

If the registration is successful, the registrar forwards requests that are destined to the address of record to the back-to-back user agent.

```
location-service INALP
  domain inalp.com

  identity-group register
    registration outbound
      register auto

  identity 400
    registration outbound
      registrar sip.inalp.com
      register auto

  identity 555 inherits register

context sip-gateway GW_SIP

  sip-interface SIP_WAN
    bind interface IF_WAN port 5060

context sip-gateway GW_SIP
  bind location-service INALP
```

If an identity is added to a location-service which is bound to a context sip-gateway, the following procedure takes place:

1. Determine if identity should be registered.
   a. The location-service containing the identity must have at least one domain configured.
   b. The identity must have a registration outbound face configured or inherited.
   c. The register command must be set to “auto”.
2. Build the address of record to register. The name of the identity builds the user-part. The first domain configured in the location-service builds the host-part.
3. Build the address of the registrar. The registrar configured in the registration outbound face is taken as request-uri. If no registrar is configured the first domain configured in the location-service builds the request-uri.
4. Build expire header. If a lifetime is configured in the registration outbound face an expire header with the desired lifetime is added.
5. Build contact address to register. The spoofed-contact parameter of the sip-interface in the context sip-gateway through which the REGISTER request is sent is set as contact address. If no spoofed-contact is configured the ip-address and port of the sip-interface in the context sip-gateway through which the REGISTER request is sent builds the contact address.
6. Send the REGISTER request.

If one of these steps has no result and fails, the registration fails. After a certain timeout (which is configurable in the registration outbound face), the request is re-issued.

Inbound Registration

With the according license, the back-to-back user agent can allow registrations from other user agents. Therefore, identities must be configured with a registration inbound face. Contacts to forward requests for this identity can be configured or added dynamically with REGISTER requests.

If the gateway has to accept register requests for unknown identities or for any identity that belongs to a certain domain, there can be a “default” identity-group configured. The configuration of the identity-group “default” with the registration inbound face allows registration for any identity in this location-service that is not explicitly configured.

```
location-service INALP
  domain inalp.com

  identity-group default
    registration inbound

  identity 400
    registration inbound
      lifetime default 4000 min 600 max 36000
      contact 172.16.40.22 switch IF_SIP priority 500

context sip-gateway GW_SIP

  sip-interface SIP_WAN
    bind interface IF_WAN port 5060

context sip-gateway GW_SIP
  bind location-service INALP
```

If the gateway receives an incoming REGISTER request, the following procedure takes place:

1. Determine to which sip interface in the context cs the request should be forwarded. This happens according the same rules as an incoming INVITE is forwarded. Outgoing calls to the registered contacts will pass through the same sip interface as the incoming REGISTER request.
2. Check request-uri. The host part of the request-uri must match a domain of a location-service which has configured imperative “authoritative” and is bound to the context sip-gateway which received the request.
3. Check to header. The host part of the to-uri must match the host part of the request-uri.
4. Check if registration is allowed. There must be an identity in the location-service that match with name or alias the host part of the to-uri or a identity-group “default” with a registration inbound face configured.
5. Create a dynamic identity if the identity to register does not already exist. This happens when the identity-group “default” is configured with a registration inbound face. The dynamic created identity inherits from the identity-group “default”.
6. Add all contacts with expires≠0 to the requested identity. The expiration time is taken out of the expire parameter from the contact or from the expire header. This time is adjusted to fit into the configured lifetime boarder. If there is no expires parameter the default expired from the configured lifetime is taken.
7. Remove all contacts with expires=0 from the requested identity.
8. Remove a dynamic identity if the identity contains no more contacts.

9. Return 200 OK with all contacts registered or configured from the requested identity

If one of these steps fails the registration fails and an according message is sent. A registered contact is removed out of the location-service after the expiration time has passed and the registration was not refreshed. The "SIP location-service" (on page 526) in the context cs is able to forward calls to the registered contacts.

B2B User Agent with Registered Clients

The sip-location-service in the context cs is used to route calls according to the contact bindings. Calls can be routed from other services, interfaces or tables to the sip-location-service. The sip-location-service routes the call directly to the sip-interface over which the according contact was registered. If there are some mappings or adjustments of call parameters needed, this has to be done before routing to the sip-location-service. The address-translation of the destined sip interface is the only mapping that happens after the sip-location-service.

If calls from an ISDN, FXS or FXO interface are routed to the sip-location-service, it is necessary to bind a location-service to the sip-location-service because the domain information is needed. An alternative way would be to map a complete sip-uri to the call destination properties.

The mode command in the sip-location-service configures the behavior of the service when multiple contacts are registered for one address of record. In "distribute" mode, the call is distributed to all contacts at the same time. The first phone that picks up receives the call. In "hunt" mode, one contact after each other is called, and the hunt-timeout parameter defines how long to wait until proceeding to the next contact. The contacts with higher priority are hunted first. In "distribute-and-hunt" mode, all contacts with the same priority are distributed together and the hunt searches from the higher priority contacts to the lower priority contacts. If no hunt-timeout is configured, there is no hunting.

```
context cs switch
  interface sip IF_SIP
    bind context sip-gateway GW_SIP
    route call dest-service SERVICE_SIP
    remote inalp.com
    local inalp.com

  interface isdn IF_ISDN
    route call dest-service SERVICE_SIP

  service sip-location-service SERVICE_SIP
    bind location-service INALP

location-service INALP
  domain inalp.com

  identity-group default
    registration inbound

context sip-gateway GW_SIP

  sip-interface SIP_WAN
    bind interface IF_WAN port 5060

context sip-gateway GW_SIP
  bind location-service INALP
```

If the sip-location-service receives a call, the following procedure takes place:

1. Determine the requested domain. If none of these three possibilities matches the call is dropped.
   a. If the call has a destination-uri set the host part of that uri is taken as requested domain.
   b. If there is no destination-uri set, but a destination-ip-address, this is taken as requested domain.
   c. If there is no destination-uri and no destination-ip-address set, but a location-service bound, the requested domain is taken from there.
2. Determine the requested user.
   a. If the call has a destination-uri set the user part of that uri is taken as requested user.
   b. If there is no destination-uri set, the destination-e164 is taken as requested user.
3. Determine the location-service. The location-services are checked if one domain matches the requested domain and if the imperative of the location-service is “authoritative”. If a location-service is bound and it does not match for the requested domain the call is dropped.
4. Determine the identity. The location-service is searched for an identity where the name or an alias matches the requested user.
5. Get the registered and configured contacts of that identity. The contacts are sorted according to the priorities.
6. Distribute or Hunt the contacts according the configured mode.

If one of these steps fails, the call fails. It is recommended to configure a hunt-group in front of that service to have a fall-back when the call fails because the requested identity is not registered.

Chapter 47 VoIP profile configuration

Chapter contents

Introduction 574
VoIP profile configuration task list 575
Creating a VoIP profile 575
Configure codecs 576
Configuring the transparent-clearmode codec 578
Configuring the Cisco versions of the G.726 codecs 578
Configuring DTMF relay 579
Configuring RTP payload types 579
Configuring RTP payload type for transparent-clearmode 580
Configuring RTP payload types for the g726-32k and g726-32k-cisco coders 580
Configuring RTP payload type for Cisco NSE 580
Configuring Cisco NSE for Fax 580
Configuring the dejitter buffer (advanced) 581
Enabling/disabling filters (advanced) 583
Configuring Fax transmission 584
T.38 CED retransmission 587
T.38 No-Signal Retransmission 588
Fax bypass method 588
Configuring fax failover 588
Configuring modem transmission 589
Modem bypass method 589
Configuring the traffic class for Voice and Fax data 590
Configuring IP-IP codec negotiation 590
Examples 591
Home office in an enterprise network 591
Home office with fax 593
Soft phone client gateway 594

Introduction

This chapter gives an overview of VoIP profiles, and describes how they are used and the tasks involved in VoIP profile configuration.

A VoIP profile is a container for all datapath-related settings on VoIP connections. The profile settings apply to all calls going through the interface. A VoIP profile can be assigned to VoIP gateways and VoIP interfaces in context CS. If no profile is specified for a particular interface, a profile from the gateway the interface binds to is used instead. Figure 1 illustrates the relations between VoIP profiles, gateways and CS interfaces. The following components are configurable:

- Codecs and codec parameters (such as silence suppression, RTP payload type, and audio filters)
- DTMF relay
- Dejitter buffer
- Fax transmission
- Modem transmission

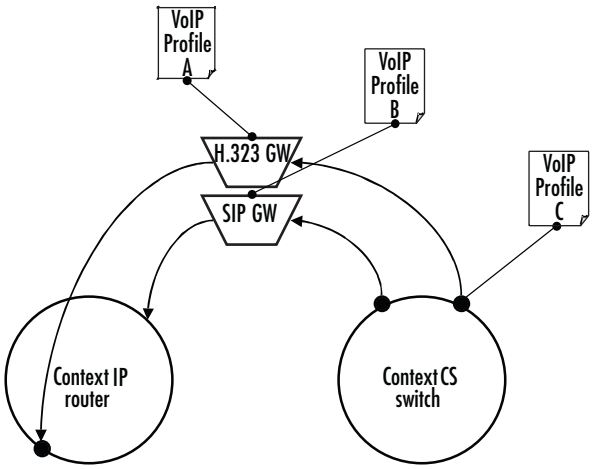


Figure 1. VoIP profile association

IMPORTANT

Configuring voice datapath options can improve ***or degrade*** the quality of the transmitted voice data. Many of the default values of these components have configured defaults that should only be changed if required. Misconfiguration can strongly affect the voice quality perceived by the user and the bandwidth requirements of VoIP connections. Be sure you understand the meaning and impact of all commands prior to changing any settings.

VoIP profile configuration task list

The following tasks describe components that can be configured through the VoIP profile:

- Creating a VoIP profile
- Configuring codecs
- Enabling DTMF relay (see page 576)
- Configuring RTP payload types (see page 579)
- Configuring the dejitter buffer (advanced) (see page 579)
- Enabling/disabling filters (advanced) (see page 583)
- Configuring fax transmission (see page 584)
- Configuring modem transmission (see page 589)

If a VoIP profile is modified, the saved modification is applied to all open calls and is valid for all future calls on the gateway or interface using this VoIP profile.

Creating a VoIP profile

Before configuring voice parameters, a VoIP profile must be created. Each VoIP profile has a name that can be any arbitrary string of not more than 25 characters. When you create the VoIP profile, the VoIP profile configuration mode appears so you can configure VoIP components.

Note The VoIP profile named *default* always exists in the system. It is used by all interface components if there is no other VoIP profile available. If VoIP parameters are the same throughout all interfaces, you can simply change the profile *default* instead of creating a new profile.

Procedure: Create a VoIP profile and enter the VoIP profile configuration mode

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#profile voip** *name*	Creates a VoIP profile with name *name* and enters VoIP profile configuration mode. The newly created profile contains default values for all parameters. If a profile with name *name* already exists, only the VoIP profile configuration mode is entered.
2	***node*(pf-voip)[*name*]#...**	Configuration steps are described in the following sections.

Example: Creating a VoIP profile

This example shows how to create a VoIP profile named g729_FaxRelay and enter VoIP profile configuration mode.

```
node>enable
node#configure
node(cfg)#profile voip g729_FaxRelay
node(pf-voip)[g729_fa~]#...
```

Configure codecs

The VoIP profile contains a *list* of codecs the forms the set of allowed codecs that can be used to set up a VoIP connection. The list is assembled in order of priority (i.e. the first entered codec is the most preferred one). For each codec in the list, a set of parameters can be configured.

Signaling protocols have a codec negotiation mechanism, it is not guaranteed that the first codec in the list is used to set up the connection. Each codec in the list may be used. To make sure that only one codec is possible, configure this codec alone. See how to display the currently configured codecs in a VoIP profile on page 589.

The default VoIP profile contains the codecs G.711uLaw and G.711aLaw. If you don't want to use these, you must explicitly remove them from the list.

Procedure: Add a codec to the list (this procedure is valid for all other codecs as well).

Note If you press the **<tab>** key after entering a few letters of a configuration command, the full command name will display or a listing of commands that begin with those letters will display. Press the **<enter>** key to select the desired command.

Mode: Profile VoIP

Step	Command	Purpose
1	***node*(pf-voip)[*name*]#codec g729 tx-length 30 rx-length 30 silence-suppression**	Appends codec g729 to the list of codecs. Specifies the payload duration for transmitted RTP packets of this codec, and the maximum supported payload duration for received RTP packets of this codec. Allows silence suppression to be used with this codec. If the codec g729 already existed in the list, its parameters are updated with the entered values. The following codecs are available: • g711alaw64k • g711ulaw64k • g723-5k3 • g723-6k3 • g726-16k • g726-16k-cisco[a] • g726-24k • g726-24k-cisco • g726-32k • g726-32k-cisco • g726-40k • g727-16k • g727-24k • g727-32k • g729 • netcoder-6k4 • netcoder-9k6 • transparent • transparent-cisco

a. Cisco does not use the standard ITU G.726 version of G.726, but the ATM AAL2 version. This build series now supports both versions of these codecs. The Cisco G.726 codecs are available in *profile voip* as separate codecs with their name ending in *-cisco*.

Procedure: Remove a codec from the list

Mode: Profile VoIP

Step	Command	Purpose
1	***node*(pf-voip)[*name*]#no codec g729**	Remove codec g729 from the list of codecs.

Procedure: Insert a codec at a specific position in the list

Mode: Profile VoIP

Step	Command	Purpose
1	***node*(pf-voip)[*name*]#codec 1 g729 tx-length 30 rx-length 30 silence-suppression**	Inserts codec g729 at the first position of the list (most preferred codec). The parameters are the same previously described. If the codec g729 had yet existed in the list, it is moved to the first position of the list, adopting the entered parameter values.

Configuring the transparent-clearmode codec

To be compatible with RFC4040, transparent-clearmode was made available as a codec in the voip profile. The codec can be used if exclusively packetisation and no coding/decoding is needed.

Mode: configure/profile voip

Step	Command	Purpose
1	**[*name*](pf-voip)[*profile*]#[no] codec transparent-clearmode**	Allows to use the codec transparent-clearmode.

Configuring the Cisco versions of the G.726 codecs

The Cisco versions of codecs are listed in the previous section as separate codecs with their name ending in –cisco. SmartWare supports four Cisco codec versions: *g726-16k-cisco*, *g726-24k-cisco*, *g726-32k-cisco*, and *transparent-cisco*. Three of the codecs are variations of G.726, the fourth is *transparent-cisco*.

The *transparent-cisco* codec provides full compatibility with Cisco's *clear-channel* codec used for transmission of Unrestricted Digital Information over a VoIP (SIP or H.323) network.

Cisco does not use the standard ITU G.726 version of G.726, instead it uses the ATM AAL2 version.

All supported Cisco codecs are available in profile voip.

Mode: VoIP *name*

Step	Command	Purpose
1	***node*(pf-voip)[*name*]#codec { g726-16k-cisco | g726-24k-cisco | g726-32k-cisco | ... }**	To operate with Cisco's G.726 codecs.

The next table indicates the method of configuring a Cisco-variant codec as the most preferred codec. This example sets the 'transparent-cisco' as number 1, the most preferred.

Mode: VoIP *name*

Step	Command	Purpose
1	**node(pf-voip)[*name*]#codec 1 transparent-cisco**	Configures transparent-cisco as the most preferred codec.

Configuring DTMF relay

Dual tone multi-frequency (DTMF) tones are usually transported accurately in band when using high bit-rate voice codecs such as G.711. Low bit-rate codecs such as G.729 and G.723.1 are highly optimized for voice patterns and tend to distort DTMF tones. The **dtmf relay** command solves the problem of DTMF distortion by transporting DTMF tones out-of-band or separate from the encoded voice stream as shown in figure 2. H.323 signals the DTMF tones as H.245 user input indications, SIP uses a mechanism of RTP to reliably transport tones (according to RFC2833).

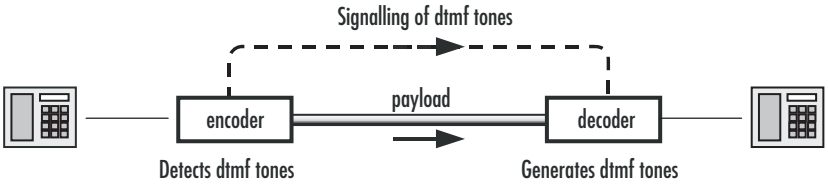


Figure 2. DTMF Relay

This procedure describes how to configure DTMF relay

Mode: Profile VoIP

Step	Command	Purpose
1	***node*(pf-voip)[*pf-name*]#dtmf-relay {default|rtp|signaling}**	Define how DTMF digits shall be transmitted. You can configure the SIP gateway to send DTMF digits either by using the SIP INFO method or in the RTP stream (RFC 2833). The gateway will always receive both types of DTMF digits. The command has currently no effect for H.323 calls.

Configuring RTP payload types

If you are using DTMF relay with SIP, the DTMF digits are transported in RTP packets with a special payload type. The default value for this payload type can be configured in the profile VoIP.

Procedure: Configure RTP NTE payload type

Mode: Profile VoIP

Step	Command	Purpose
1	***node*(pf-voip)[*name*]#rtp payload-type nte** *payload-type*	Specifies the RTP payload-type for named tone events NTE (RFC2833). Default: 101.

Configuring RTP payload type for transparent-clearmode

The following command configures the RTP payload type used for the transparent-clearmode codec.

Mode: configure/profile voip

Step	Command	Purpose
1	[*name*](**pf-voip**)[*profile*]#**[no] crtp payload-type transparent-clearmode <payload-type>**	Specifies the rtp payload type used for transparent clearmode. Value must be between 97 and 127. Default value is 97.

Configuring RTP payload types for the g726-32k and g726-32k-cisco coders

The following command specifies the RTP payload types for the g726-32k and the g726-32k-cisco coders to be used. It allows changing the payload types to a value in the range of 96 to 127 whereas the default value is 2 for both. Once a payload type has been changed, the 'no' form of the command must be used to go back to the default value.

Mode: profile voip

Step	Command	Purpose
1	[*name*](**pv-voip**)[*name*]#**[no] rtp payload-type [g726-32k | g726-32k-cisco] <value 96..127>**	Defines the RTP payload types for the g726-32k and the g726-32k-cisco coders. Default: 2

Configuring RTP payload type for Cisco NSE

Configure the RTP payload-type when transmitting the NSE events. This payload-type is negotiated during call-setup when using SIP.

Named Service Events (NSE) are the Cisco-proprietary version of Named Telephony Events (NTEs). NTEs are defined in RFC 2833. Various telephony signaling events use tones, for example, DTMF. NSEs and NTEs communicate these tones (for representing signaling events), not by the presence of tones, but by sending a binary code representing the tone that is recreated at the destination. Cisco's proprietary NSEs use different values to represent tones and events than the NTEs use.

NSEs are normally sent with RTP payload type 100. The RTP packets have the same source and destination IP addresses and UDP ports as the other packets in the media stream, but differ in the RTP payload types so they can be distinguished from the stream's audio packets.

Mode: Profile VoIP

Step	Command	Purpose
1	**node(pf-voip)[** *name*]**#rtp payload-type nse** payload-type	Specifies the RTP payload-type for Named Signaling Events (NSE). Default: *100*

Configuring Cisco NSE for Fax

This command specifies the method to be used for signaling the remote device the RTP Stream has switched to a voice-band Modem transmission. This feature is only available on the SIP protocol. If the command option

'v150-vbd' is selected, a re-invite will be sent even if the current voice coder is configured the same as the modem bypass coder. Furthermore the re-invite contains a gpmd-attribute line with the value 'vbd=yes;ecan=off ' in the media description part. This attribute signals the remote device of the new media transmission. If the command option 'default' is selected, the system behavior is the same as before.

SmartWare also supports the Cisco NSE standard which uses RFC2833 events for modem transmission over a VoIP (SIP or H.323) network. Upon detecting a modem transmission, the called peer issues NSE Event 192. NSE Event 192 indicates a Voice Band Data stream that forces the calling peer to deactivate Voice Activity Detection and to reconfigure the de-jitter buffer for data reception. Afterwards it issues the NSE Event 193 to trigger the calling peer to switch off Echo-Cancellation.

Configuring the dejitter buffer (advanced)

Packet networks always introduce a certain amount of jitter in the arrival of voice packets. To compensate for the fluctuating network conditions, a dejitter buffer is integrated in the RTP processing engine. Typical voice sources generate voice packets at a constant rate, the matching voice decompression algorithm also expects incoming voice packets to arrive at a constant rate. However, the packet-by-packet delay inflicted by the network may be different for each packet. As shown in figure 3, the result of the delays is that packets which are sent equally spaced from the left-hand gateway arrive irregularly spaced at the right-hand gateway.

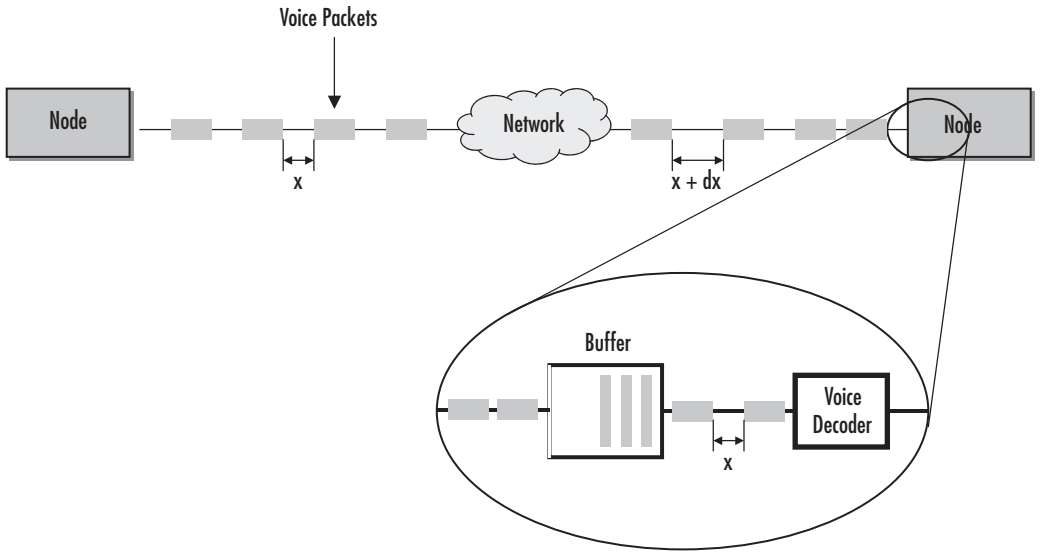


Figure 3. Jitter and dejitter buffer

The dejitter buffer delays incoming packets so it can present them to the decompression algorithm at fixed intervals. It will also fix any out-of-order packets by looking at the sequence number in the RTP packets. Such buffering has the effect of smoothing packet flow, and increasing the resiliency of the codec to packet loss, delayed packets, and other transmission effects. The negative side of dejitter buffering is that it can add significant delay. The dejitter buffer size is configurable and can be optimized for given network conditions, the size is usually set to be a multiple of the expected packet arrival time in order to buffer an integral number of packets. It is not uncommon to see dejitter buffer settings around 80 ms for each direction.

The operating modes for the dejitter buffer are illustrated in figure 4:

- Adaptive—The adaptive buffer automatically adapts to variations in the network's delay characteristics and in general yields the best results for voice conversations.

IMPORTANT

In the adaptive dejitter buffer there are parameters that can be configured (such as shrink-speed, grow-step, etc.) that should not be changed unless it is necessary to do so. An incorrect configuration can lead to interoperability problems and loss of service. ***Therefore, it is strongly recommended that only experienced users change these parameters.***

- Static—The static buffer is useful for voice conversations if you have specific information about your network's delay characteristics (such as jitter period, etc.), so it should only be used by experienced users.
- Static-data—The static-data mode if you want to create a profile for fax or modem transmission without using the T.38 or fax bypass features described later in this chapter

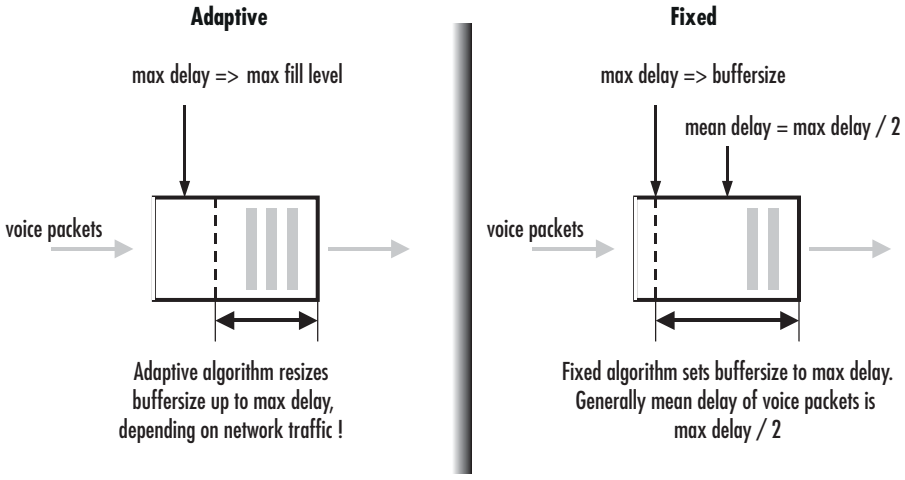


Figure 4. Adaptive versus static dejitter buffer

Procedure: Configure the dejitter buffer.

Mode: Profile VoIP

Step	Command	Purpose
1	***node*(pf-voip)[*name*]#dejitter-mode** *mode*	Specify the dejitter buffer as adaptive, static or static-data.
2	***node*(pf-voip)[*name*]#dejitter-max-delay** *max-delay*	Specify the maximum delay in milliseconds that the dejitter buffer is allowed to introduce. This setting is valid for all modes.

Enabling/disabling filters (advanced)

The voice decoder output is normally filtered through a perceptual post-filter to improve voice quality. Likewise a high pass filter is normally used to cancel noises at the coder input. When the communication channels include several SmartNodes in tandem as shown in figure 5, sequential post filtering or high pass filtering can cause degrade signal quality. In this case, the user can choose to disable the post-filter and the high-pass-filter.

Note Filtering only occurs with G.723 and G.729 codecs.

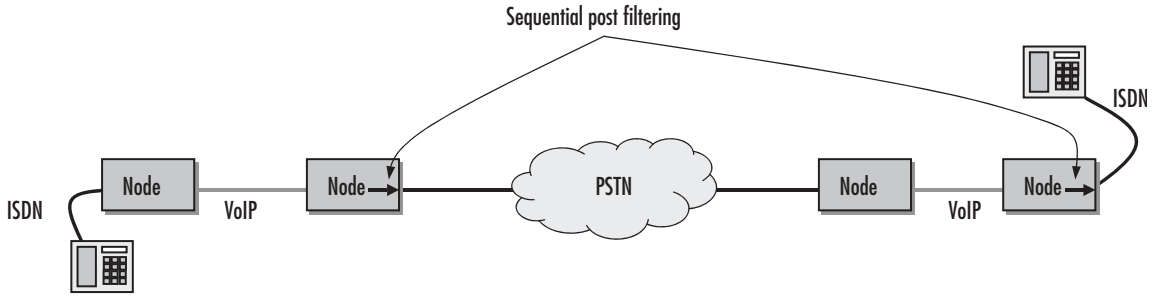


Figure 5. Multiple tandem and sequential post filtering

This procedure describes how to disable post-filtering and high-pass-filtering.

Mode: Profile VoIP

Step	Command	Purpose
1	***node*(pf-voip)[*name*]#no post-filter**	Disable decoder output filter
2	***node*(pf-voip)[*name*]#no high-pass-filter**	Disable decoder input high pass filter

Example: Disable filters

The following example shows how to disable the decoder output post-filter and the input high-pass filter.

```
node>enable
node#configure
node(cfg)#profile voip myProfile
node(pf-voip)[myProfi~]#no post-filter
node(pf-voip)[myProfi~]#no high-pass-filter
```

Configuring Fax transmission

Fax is a protocol for electronically transmitting written material in-band over a voice channel. In public switched telephone networks (PSTN), a fax is handled the same way as a voice conversation. A G3 Fax device transforms (modulates) a scanned page into audible tones that are transmitted in-band. The receiving device converts the tones (demodulates) and reconstructs the page. In IP networks, problems can make it difficult to handle a faxed call in the same way as a voice call:

- If one or more RTP packets that transport the voice (tones) are lost, the receiver can't reconstruct what the sender sent.
- Codecs other than G.711 compress the voice streams. They are optimized for compressing voice and not modulated data. Compressing and decompressing always incurs a loss of data.

SmartWare provides two solutions for fax transmission problems:

- Fax bypass—When a fax transmission is detected by the SmartNode, it automatically switches to a configured fallback codec that does no or little compression. The dejitter buffer is configured with settings optimized for fax transmission.
- Fax relay—Terminates the fax protocol on the SmartNode and sends the reference data over a fax protocol (T.38) to the receiver. Fax relay has a smaller bit-error-rate than bypass.
- Fax failover—When using fax transmission in SIP, you can configure the SIP gateway first to try T.38, but if the remote gateway does not support T.38, it will automatically fall back to a high-rate codec.

Both solutions require changing codecs during an established call, which imposes several requirements on the signaling protocol and the remote gateway. Make sure these requirements are met when configuring a fax transmission mode.

Figure 6 illustrates the difference between Fax relay and Fax bypass.

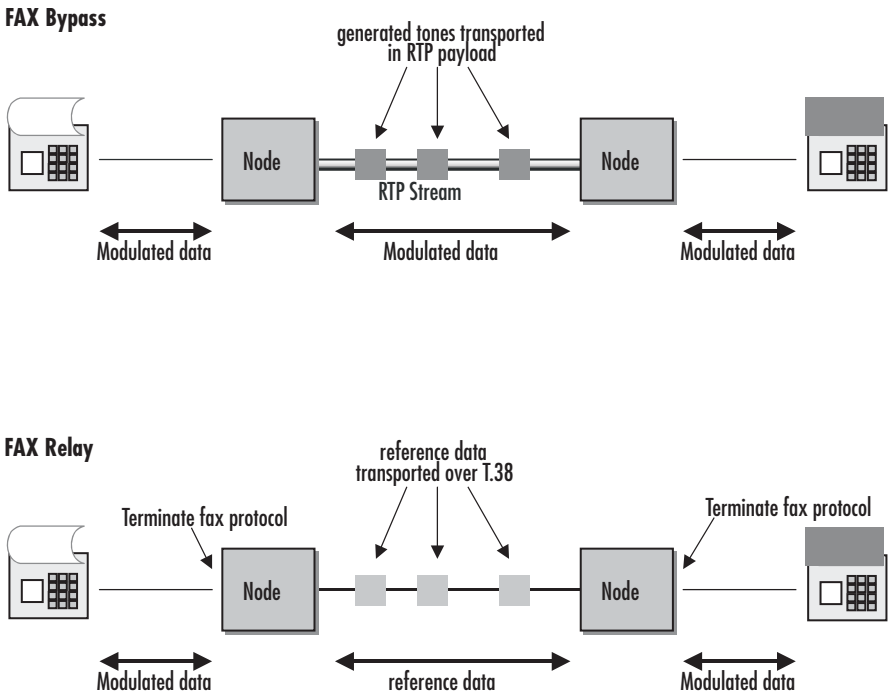


Figure 6. Fax relay and Fax bypass

Fax transmission modes are organized the same way codecs are: there is an ordered list of fax transmission modes; the most preferred fax transmission mode is the first one in the list.

Procedure: Configure fax bypass

Mode: Profile VoIP

Step	Command	Purpose
1	***node*(pf-voip)[*name*]#fax transmission bypass g711alaw64k**	Adds fax bypass transmission with codec G.711 to the list of fax transmission modes.Alternative codecs available are: • G.711uLaw • G.726 32kbps • G.726 24kbps.
2 optional	***node*(pf-voip)[*name*]#fax dejitter-max-delay** *buffer-size*	Sets the size of the dejitter buffer during fax transmissions. The operating mode of the dejitter buffer is automatically set to fax optimized static-data mode. Patton recommends that you keep the size for fax transmissions higher than that used for voice, since fax is less sensitive to delay than packet loss. The default value is 200ms which should be nominal for almost any transmission network. In exceptional cases it may be necessary to increase this value (maximum 400ms).

Procedure: Configure fax relay (T.38)

Mode: Profile VoIP

Step	Command	Purpose
1	***node*(pf-voip)[*name*]# fax transmission relay t38-udp**	Adds fax relay transmission with T.38 protocol over UDP to the list of fax transmission modes.
2 (optional)	***node*(pf-voip)[*name*]#fax redundancy ls** *low-speed-redundancy* **hs** *high-speed-redundancy*	Packet loss can be avoided by transmitting the fax data packets several times. This can be configured separately for low speed and the high speed traffic. The default for both parameters is 0 (no redundant transmission). Note that values greater than 0 provide more reliable transmissions, but consume additional bandwidth.
3 (optional)	***node*(pf-voip)[*name*]# fax dejitter-max-delay** *buffer-size*	For proper operation, a dejitter buffer is used on the receiver. The dejitter period can be set to compensate for the jitter imposed by the network. The default value is 200ms which should be nominal for almost any transmission network. Only in exceptional cases it may be necessary to increase this value (maximum 400ms). The dejitter buffer, by default, applies the operation mode 'static-data', i.e. minimizes the packet loss.

Step	Command	Purpose
4 (optional)	***node*(pf-voip)[*name*]#fax volume** *volume*	Adjusts the volume of the fax signals re-generated on the receiver side. The volume is in dB, in the range -18.5 ... -3.5 (Default: -9.5dB).
5 (optional)	***node*(pf-voip)[*name*]# fax max-bit-rate { 2400 | 4800 | 7200 | 9600 | 12000 | 14400 }**	Sets maximum allowed bit-rate for fax relay (Default 14400 Bit/sec).
6 (optional)	***node*(pf-voip)[*name*]# fax detection { ced-tone | fax-frames }**	Selects the method when fax transmissions are detected: By CED tone or by fax frames (Default: ced-tone). It takes longer to detect Fax frames than CED tones, but the risk of misdetection is minimized.
7 (optional)	***node*(pf-voip)[*name*]#no fax error-correction**	Disables error correction mode (Default: enabled). If the error correction mode is disabled, the connected fax devices cannot negotiate error correction mode. Connections with error correction mode enabled are more sensitive to packet loss. Disable error correction mode when packet loss is more than 2–3%. **Note** Error correction mode does not cancel IP packet loss.
8 (optional)	***node*(pf-voip)[*name*]#no fax hdlc**	Disables HDLC image transfer (Default: enabled). If HDLC mode is enabled, the SmartNode removes bit-stuffing, checks CRCs of fax frames arriving from the PSTN and regenerates the CRCs before sending fax frames towards the PSTN. HDLC can only be enabled together with error correction. Disable HDLC when the fax peer does not support this mode.

T.38 CED retransmission

Even if the user has configured redundant transmission for low-speed and high-speed packets the T.30 Indicator messages are not included in this process. If the CED message gets lost, the remote device only receives the CED Tone that is sent in-band. But the transmitted in-band tone may be short due to T.38 switchover or too much distortion, such as by using a low bit-rate voice coder like G.723. In this case it is possible that the remote device never starts the initial T.30 procedure, because it has never received the CED tone. For that reason SmartWare sends the CED three times in an interval of 100ms with the same sequence number. With this command, you can disable this feature or set the number of retransmissions to a user defined value.

Mode: profile voip *profile-name*

Step	Command	Purpose
1	[*name*] **(pf-voip)**[name]**#[no] fax ced-retransmission** *number*	Specifies the number of CED retransmissions. Default: *2*

T.38 No-Signal Retransmission

Some SIP gateways change their port number when switching from audio to T.38. This behavior causes problems if the SmartNode is located on the A-Side behind a NAT. Due to T.30 is a unidirectional. protocol and the B-Side is normally the initiator of the T.30 handshaking, the SmartNode never receives the initial packets of the B-Side because the NAT ports are not yet opened.

To open the NAT ports SmartWare no sends T.38 'no-signal' packets when a codec change is detected. By default SmartNode sends 3 such packets. To adjust the number of 'no-signal' packets, use the following configuration command.

Mode: Configure/profile voip

Step	Command	Purpose
1	[*name*] **(pf-voip)**[name]**#fax nosignal-retransmission [1...5]**	Sets how many times a T.38 'nosignal' is retransmitted. Default: *3*

Fax bypass method

This command specifies the method for notifying the remote device that the RTP Stream has switched to a voice-band FAX transmission. This feature is only available on the SIP protocol. If the command option '**v150-vbd**' is selected, a re-invite is sent even if the current voice coder is configured the same as the fax bypass coder. Furthermore the re-invite contains a gpmd-attribute line with the value '**vbd=yes**' in the media description part. It signals the remote device of the new media transmission. If the command option 'default' is selected, the system behavior is the same as before.

For a fax transmission over a VoIP (SIP or H.323) network, the Cisco NSE standard uses events defined by RFC2833. These events are used for the setup of the fax transmission starting between the calling- and called-peer. Upon detecting a fax transmission, the called-peer issues NSE Event 192. NSE Event 192 indicates the data stream is via a voice band, and it forces the calling-peer to do two things—deactivate voice activity detection and reconfigure the de-jitter buffer for data reception. The option 'nse' enables this fax transmission standard.

Mode: profile voip

Step	Command	Purpose
1	[*name*] **(pf-voip)**[name]**#fax bypass-method {default | v150-vbd|nse}**	Specifies the fax bypass signaling method. Default: *default*

Configuring fax failover

When using fax transmission in SIP, it is possible configure the SIP gateway to first try to use T.38 and to fall back to a high-rate codec, if the remote gateway does not support T.38. This can be configured as follows:

Mode: profile voip <pf-name>

Step	Command	Purpose
1	[*name*] **(pf-voip)**[pf-name]**# fax transmission 1 relay t38-udp**	Define T.38 UDP as the first fax transmission method to try

Step	Command	Purpose
2	[*name*] **(pf-voip)**[pf-name]**# fax transmission 2 bypass g711alaw64k**	Define G.711 A-Law as the second fax transmission method to try, if T.38 is not supported by the remote gateway.

Note The first codec must always be T.38, while the second one must be a high-rate codec such as G.711, which supports fax transmission.

Configuring modem transmission

Modem transmission is similar to fax transmission, except that modem data is always transported in bypass mode. This means that an ordered list of bypass codecs can be defined for modem transmission. If no modem transmission codec is configured, no action is taken to change the codec when modem is detected.

Procedure: Configure modem bypass

Mode: Profile VoIP

Step	Command	Purpose
1	***node*(pf-voip)**[*name*]**#modem transmission bypass g711alaw64k**	Adds modem transmission with codec G.711 to the list of fax transmission modes.Alternative codecs available are: • G.711uLaw • G.726 32kbps • G.726 24kbps.

Modem bypass method

This command specifies the method to be used for signaling the remote device the RTP Stream has switched to a voice-band Modem transmission. This feature is only available on the SIP protocol. If the command option *v150-vbd* is selected, a re-invite will be sent even if the current voice coder is configured the same as the modem bypass coder. Furthermore the re-invite contains a gpmd-attribute line with the value *vbd=yes;ecan=off* in the media description part. This attribute signals the remote device of the new media transmission. If the command option *default* is selected, the system behavior is the same as before.

SmartWare also supports the Cisco NSE standard which uses RFC2833 events for modem transmission over a VoIP (SIP or H.323) network. Upon detecting a modem transmission, the *called* peer issues NSE Event 192. NSE Event 192 indicates a Voice Band Data stream that forces the *calling* peer to deactivate Voice Activity Detection and to reconfigure the Dejitter Buffer for data reception. Afterwards it issues the NSE Event 193 to trigger the calling peer to switch off Echo-Cancellation.

Mode: profile voip

Step	Command	Purpose
1	[*name*] **(pf-voip)**[name]**#modem bypass- method {default | v150-vbd | nse}**	Specifies the modem bypass signaling method. Default: *default*

Configuring the traffic class for Voice and Fax data

The traffic class for voice data and fax data is configurable. The configured traffic class is used as additional routing criterion in the IP routing table.

Mode: Profile VoIP

Step	Command	Purpose
1	[*name*] **(pf-voip)[***profile***]# rtp trafficclass <traffic-class>**	Sets traffic class for voice data and fax data packets. The traffic class may be new or may already exist.

Configuring IP-IP codec negotiation

This command is only available on the SN4960 and applies to calls between two IP endpoints (e.g. SIP-SIP or H.323-SIP). It is in mode “profile voip”.

Step	Command	Purpose
1	***node*(pf-voip)[***default***]#[no] codec negotiation**	Enables/disables codec negotiation.

Disabled “codec negotiation” honors the codec lists from each call leg independently, formed out of the remote and local capabilities. The DSP is inserted into the RTP path to make sure each side can use its codec. The DSP is transcoding between the codecs of the two RTP streams.

Enabled “codec negotiation” will keep the DSP out of the picture for IP-IP calls and tries to negotiate a common codec for both call legs.

Examples

Different applications require different VoIP profiles. This section includes a variety of applications and show how the VoIP profile for these applications would be configured.

Home office in an enterprise network

Figure 7 is an example of a home office in an enterprise network. The connection bandwidth is 128 kbps and is of very low quality, so the low bit-rate G.723_6k3 codec is used. Likewise, silence suppression is enabled. Because of the low bit-rate codec, DTMF relay is also enabled. As 80 to 100 ms jitter is anticipated, the dejitter buffer is set to *adaptive* with a maximum delay of 100 ms.

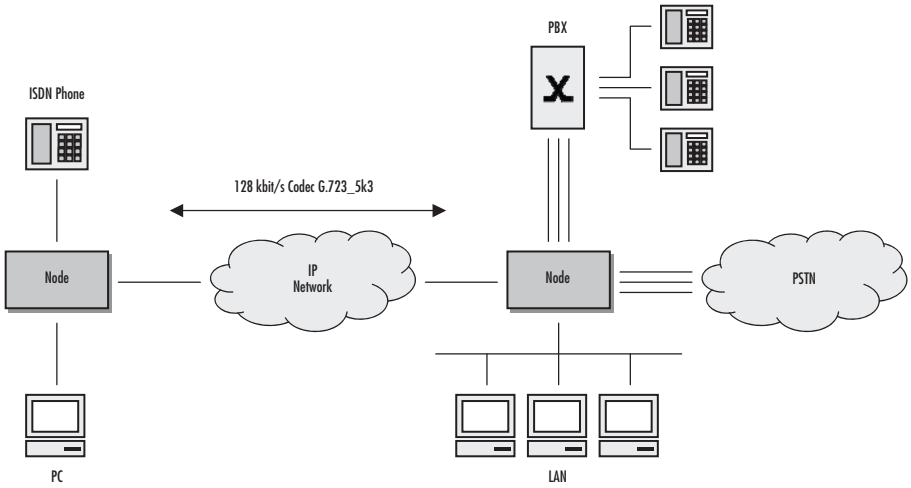


Figure 7. Home office in an enterprise network

First, configure the required CS interfaces (see chapter 33, "CS interface configuration" on page 381) and call routing (see chapter 40, "Call router configuration" on page 456).

Next, configure the voice over IP settings as needed based on the previous description. First we create the VoIP profile with the needed configurations.

```
1 node>enable
2 node#configure
3 node(cfg)#profile voip Wire128kbit
4 node(pf-voip)[Wire128~]#no codec g711aLaw64k
5 node(pf-voip)[Wire128~]#no codec g711uLaw64k
6 node(pf-voip)[Wire128~]#codec g723-6k3 tx-length 30 rx-length 30 silence-suppres-
sion
7 node(pf-voip)[Wire128~]#dejitter-max-delay 100
8 node(pf-voip)[Wire128~]]#show profile voip Wire128kbit
VoIP Profile: Wire128kbit
=========================

 Used:                                  by 0 module(s)

 Codecs
 ------
```

```
 G.723 6k3:                          rxlen=30;txlen=30;ss

Fax Transmission
Modem Transmission

Dejitter
--------

 Mode:                               Adaptive
 Max. Delay:                         100ms
 Max. Packet Loss:                   4/1000
 Shrink Speed:                       1
 Grow Step:                          1
 Grow Attenuation:                   1
 High Pass Filter:                   enabled
 Post Filter:                        enabled

Fax
---

 Detection:                          CED Tone
 T.38 High Speed Redundant Packets:  0
 T.38 Low Speed Redundant Packets:   0
 Max. Bit Rate:                      14400bps
 Volume:                             -9.500dB
 Error Correction:                   enabled
 HDLC:                               enabled
 Dejitter Max Delay:                 200ms

Modem
-----

 Max. Bit Rate:                      14400
 Volume:                             -9.500dB
 HDLC:                               enabled

DTMF
----

 Relay:                              enabled
 Mute Encoder:                       enabled

RTP
---

 Payload Type NTE:                   101
```

Description:

3. Create VoIP profile and give it a name. All settings have default values

4., 5. Remove the default codecs G.711alaw and G.711uLaw

6. Add codec g723-6k3 with silence-suppression enabled

7. Allow the dejitter buffer to compensate 100 milliseconds of network jitter.

8. Show the configured profile.

Home office with fax

Preconditions are those used in section "Home office in an enterprise network" on page 591: low bandwidth and high jitter. In this example, bandwidth is 256 kbps, what enables us to use the G.729 codec. But since the fax protocol must also be supported, the configuration is extended:

```
1 node>enable
2 node#configure
3 node(cfg)#profile voip g729_FaxRelay
4 node(pf-voip)[g729_Fa~]#no codec g711aLaw64k
5 node(pf-voip)[g729_Fa~]#no codec g711uLaw64k
6 node(pf-voip)[g729_Fa~]#codec g729 tx-length 20 rx-length 20 silence-suppression
7 node(pf-voip)[g729_Fa~]#dejitter-max-delay 100
8 node(pf-voip)[g729_Fa~]#fax transmission relay t38-udp
9 node(pf-voip)[g729_Fa~]#fax max-bit-rate 9600
10 node(pf-voip)[g729_Fa]]#show profile voip g729_FaxRelay
VoIP Profile: g729_FaxRelay
===========================

 Used:                                  by 0 module(s)

 Codecs
 ------

  G.729A:                               rxlen=20;txlen=20;ss
  T.38 UDP

 Fax Transmission
 ----------------

  T.38 UDP

 Modem Transmission

 Dejitter
 --------

  Mode:                                 Adaptive
  Max. Delay:                           100ms
  Max. Packet Loss:                     4/1000
  Shrink Speed:                         1
  Grow Step:                            1
  Grow Attenuation:                     1
  High Pass Filter:                     enabled
  Post Filter:                          enabled

 Fax
 ---

  Detection:                            CED Tone
  T.38 High Speed Redundant Packets:    0
  T.38 Low Speed Redundant Packets:     0
  Max. Bit Rate:                        9600bps
  Volume:                               -9.500dB
```

```
 Error Correction:                    enabled
 HDLC:                                enabled
 Dejitter Max Delay:                  200ms

Modem
-----

 Max. Bit Rate:                       14400
 Volume:                              -9.500dB
 HDLC:                                enabled

DTMF
----

 Relay:                               enabled
 Mute Encoder:                        enabled

RTP
---

 Payload Type NTE:                    101
```

Description:

3. Create VoIP profile and give it a name. All settings have default values

4., 5. Remove the default codecs G.711alaw and G.711uLaw

6. Add codec g729 with silence-suppression enabled

7. Allow the dejitter buffer to compensate 100 milliseconds of network jitter.

8. Enable fax relay over T.38 protocol

9. Limit the maximum bit rate the fax devices can communicate with each other to 9600 kbps

10. Show the configured profile.

Soft phone client gateway

A soft phone client can only use G.711uLaw or G.723 codes, neither of which can use silence suppression, DTMF relay, or fax.

```
1 node>enable
2 node#configure
3 node(cfg)#profile voip softPhone
4 node(pf-voip)[softPho~]#no codec g711aLaw64k
5 node(pf-voip)[softPho~]#codec g723-6k3 tx-length 30 rx-length 30 no-silence-sup-
pression
6 node(pf-voip)[softPho~]#no dtmf-relay
7 node(pf-voip)[softPho]]#show profile voip softPhone
VoIP Profile: softPhone
=======================

 Used:                                by 0 module(s)

 Codecs
 ------
```

```
 G.711 u-law:                          rxlen=20;txlen=20
 G.723 6k3:                            rxlen=30;txlen=30

Fax Transmission
Modem Transmission

Dejitter
--------

 Mode:                                 Adaptive
 Max. Delay:                           60ms
 Max. Packet Loss:                     4/1000
 Shrink Speed:                         1
 Grow Step:                            1
 Grow Attenuation:                     1
 High Pass Filter:                     enabled
 Post Filter:                          enabled

Fax
---

 Detection:                            CED Tone
 T.38 High Speed Redundant Packets:    0
 T.38 Low Speed Redundant Packets:     0
 Max. Bit Rate:                        14400bps
 Volume:                               -9.500dB
 Error Correction:                     enabled
 HDLC:                                 enabled
 Dejitter Max Delay:                   200ms

Modem
-----

 Max. Bit Rate:                        14400
 Volume:                               -9.500dB
 HDLC:                                 enabled

DTMF
----

 Relay:                                disabled
 Mute Encoder:                         disabled

RTP
---

 Payload Type NTE:                     101
```

Description:

3. Create VoIP profile and give it a name. All settings have default values

4. Remove the default codec G.711alaw that is not supported.

5. Add codec g723-6k3 without silence-suppression

6. Disable DTMF relay.
7. Show the configured profile.

Chapter 48 PSTN profile configuration

Chapter contents

Introduction 598
PSTN profile configuration task list 598
Creating a PSTN profile 598
Configuring the echo canceller 599
Configuring output gain 599

Introduction

This chapter gives an overview of PSTN profiles, and describes how they are used and the tasks involved in PSTN profile configuration.

A PSTN profile is a container for all datapath-related settings on PSTN connections. It can be assigned to PSTN interfaces in context CS. If no profile is specified in a particular interface, the profile *default* is used. The settings apply to all calls crossing the interface. Figure 8 illustrates the relationship between PSTN profiles and CS interfaces. The following components are configurable:

- Echo canceller
- Output gain

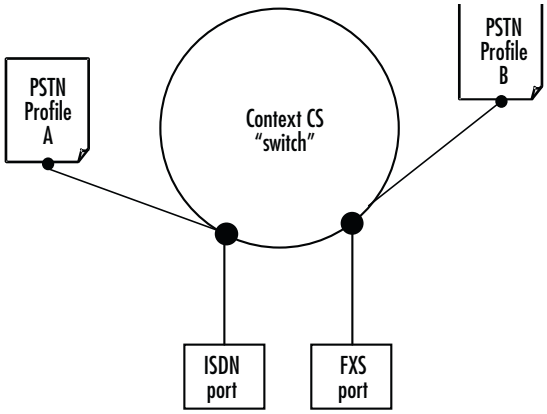


Figure 8. PSTN profile association

PSTN profile configuration task list

The following tasks describe components that can be configured through the PSTN profile.

- Creating a PSTN profile
- Configuring the echo canceller (see page 599)
- Configuring output gain (see page 599)

If a PSTN profile is modified, the saved modification is applied to all open calls and is valid for all future calls on the interface using this PSTN profile.

Creating a PSTN profile

Before configuring voice parameters, a PSTN profile must be created. Each PSTN profile has a name that can be any arbitrary string of not more than 25 characters. When you create the PSTN profile, the PSTN profile configuration mode appears so you can configure PSTN components.

Note The PSTN profile named *default* always exists in the system. It is used by all interface components if there is no other PSTN profile available. If PSTN parameters are the same throughout all interfaces, you can simply change the profile *default* instead of creating a new profile.

Procedure: Create a PSTN Profile and enter the PSTN profile configuration mode

Mode: Configure

Step	Command	Purpose
1	***node*(cfg)#profile pstn** *name*	Create a PSTN profile with name *name* and enter PSTN profile configuration mode. The newly created profile contains default values for all parameters. If a profile with name *name* already exists, only the PSTN profile configuration mode is entered.
2	***node*(pf-pstn)[*name*]#...**	Configuration steps as described in the chapters below

Configuring the echo canceller

Echoes are reflections of the transmitted signal that result from impedance mismatches in the hybrid (bi-directional 2-wire to 4-wire conversion) device, causing an echo on the wire. Echo cancellation provides near-end echo compensation for this effect as shown in figure 9.

Figure 9. Echo Cancellation

Procedure: Disable echo cancellation.

Mode: Profile PSTN

Step	Command	Purpose
1	***node*(pf-pstn)[*name*]#no echo-canceller**	Disable echo canceller (Default: Enabled)

Configuring output gain

The output gain determines the voice output volume gain towards PSTN ports as shown in figure 10.

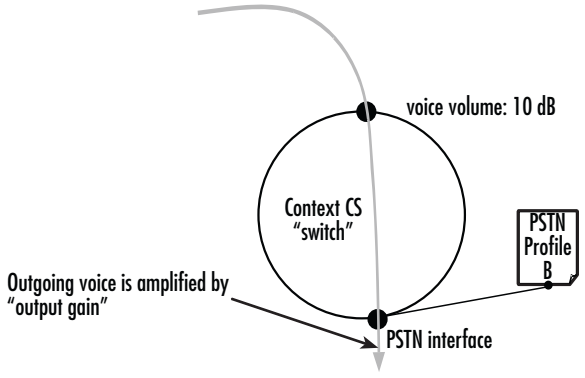


Figure 10. Applying output gain

Procedure: Configure voice output gain.

Mode: Profile PSTN

Step	Command	Purpose
1	***node*(pf-pstn)[*name*]#output-gain** *gain*	Set the output gain to value in dB

Chapter 49 SIP profile configuration

Chapter contents

Introduction 602
SIP profile configuration task list 602
Entering the configuration mode for a SIP profile 602
Mapping from a SIP disconnect cause 602
Mapping to a SIP cause 603
Mapping from a SIP redirection reason 603
Mapping to a SIP redirection code 603

Introduction

The SIP profile specifies disconnect cause mappings from SIP codes to Q.931 causes, and vice versa. As for all profiles, there is a *default* profile at system startup that can be modified. Only those causes that differ from the default mapping have to be configured. If a new profile is created, all mappings are set to their default and are only overwritten if configured. The default mapping in both directions is according to RFC3398 - ISUP to SIP Mapping.

A SIP profile can either be attached to SIP interfaces or to identities. To see how to configure a SIP profile for an interface, see Chapter 39, "SIP interface configuration" on page 441. For information about SIP profile configuration for identities, see Chapter 51, "Location Service" on page 607.

SIP profile configuration task list

This section describes the configuration tasks for SIP profile listed below.

- Enter the configuration mode for a SIP profile (see page 602)
- Map *from* a SIP disconnect cause *to* a Q.931 cause (see page 602)
- Map *to* a SIP cause *from* a Q.931 disconnect cause (see page 603)
- Map *from* a SIP redirection code *to* a Q.931 redirect reason (see page 603)
- Map *to* a SIP redirection code *from* a Q.931 redirect reason (see page 603)

Entering the configuration mode for a SIP profile

The **profile sip** command enters the configuration mode of an existing profile or creates a new one with a specified name. It also destroys an existing profile except the default, which always exists.

Mode: Configure

Step	Command	Purpose
1	**[name](cfg)#[no] profile name** *<name>*	Creates/Destroys a SIP profile or enter the configuration mode of an existing one.

Mapping from a SIP disconnect cause

The **map cause from-sip** command maps a specific SIP disconnect cause to a Q.931 cause used by the call control. All causes are pre-defined in the system and are provided by the command.

Mode: Profile SIP

Step	Command	Purpose
1	**[name](pf-sip)[name]#map cause from-sip** *sip-cause* **to** *q931-cause*	Maps a specific SIP disconnect cause to a Q.931 cause.

Mapping to a SIP cause

The **map cause to-sip** command can be used to map a call control Q.931 cause to a SIP cause. All causes are pre-defined in the system and are provided by the command.

Mode: Profile SIP

Step	Command	Purpose
1	**[name](pf-sip)[name]#map cause to-sip** *q931-cause* **to** *sip-cause*	Maps a specific Q.931 disconnect cause to a SIP cause code.

Mapping from a SIP redirection reason

The **map redir-reason from-sip** command can be used to map a specific SIP redirect code to a Q.931 redirect reason used by the call control. All redirect codes and reasons are pre-defined in the system and are provided by the command.

Mode: Profile SIP

Step	Command	Purpose
1	**[name](pf-sip)[name]#map redir-reason from-sip** *code* **to** *reason*	Maps a SIP redirection code to a Q.931 redirection reason.

Mapping to a SIP redirection code

The **map redir-reason to-sip** command can be used to map a Q.931 redirect reason to a specific SIP redirect code. All redirect codes and reasons are pre-defined in the system and are provided by the command.

Mode: Profile SIP

Step	Command	Purpose
1	**[name](pf-sip)[name]#map redir-reason to-sip** *reason* **to** *code*	Maps a Q.931 redirect reason to a SIP redirect code.

Chapter 50 **Authentication Service**

Chapter contents

Introduction 605
Authentication Service configuration task list 605
Creating an Authentication Service 605
Configuring a Realm 606
Configuring the authentication protocol 606
Creating credentials 606
Configuration Examples 606

Introduction

This chapter describes how to configure authentication services in SmartWare. The *Authentication Service* is a data base that manages *Authentication Credentials* of one or more *Realm*. A *Realm* is an *Authentication Zone* or *Authentication Domain* that defines the authentication responsibility in a network. Each *Authentication Credential* created in an Authentication Service belongs to the defined Realm and exists on a User Name and an optional Password. It is also possible to create an Authentication Service without specifying a Realm, which represents the default realm. Whenever authentication is required and the provided Realm doesn't exist in an Authentication Service, this default realm will be considered to find the right Authentication Credentials.

Authentication Service configuration task list

The following section describes how to create a new authentication service and how to enter the configuration mode of an existing service. Additionally, it describes all commands and sub commands of the authentication service configuration mode. All configuration tasks for Authentication Services are listed below.

- Create an Authentication Service (see page 605)
- Configure a Realm (see page 606)
- Configure the authentication protocol (see page 606)
- Create credentials (see page 606)

Creating an Authentication Service

The **authentication-service** command enters the configuration mode of an existing authentication or creates a new one if the requested service does not yet exist. The *no* form of the command destroys the authentication service.

Mode: Configure

Step	Command	Purpose
1	[*node*]**(cfg)# [no] authentication-service** *<name>*	Creates/Destroys an authentication service or enters configuration mode.

Configuring a Realm

The following commands add a new Realm to the authentication service. If more than one Realm has to be entered, the order of the list can be modified by using the *index* and/or *before* and *after* keywords. The no form of the command removes an existing Realm from the list.

Mode: Authentication Service

Step	Command	Purpose
1	[*node*](ls)[name]# [no] realm *<name>* or [*node*](ls)[name]#realm *<index> <name>* or [*node*](ls)[name]#realm before *<index> <name>* or [*node*](ls)[name]#realm after *<index> <name>*	Adds or removes a Realm to/from the authentication service.

Configuring the authentication protocol

The **protocol** command specifies the authetnication protocol.

Mode: Authentication Service

Step	Command	Purpose
1	[*node*](ls)[name]#protocol {http}	Specifies the authentication protocol to be used.

Creating credentials

The following command creates Authentication Credentials identified by the entered username. The no form of the command removes an existing Credential. It is possible to enter this command without a password.

Mode: Authentication Service

Step	Command	Purpose
1	[*node*](ls)[name]# [no] username *<user>* [password *<password>*]	Creates or removes authentication credentials.

Configuration Examples

```
authentication-service AUTH_SRV
  realm 1 voip-public
  realm 2 voip-intranet
  realm 3 ms-exchange
  username 433 password fK+bfnzL45Goh/VdjrWxAA== encrypted
  username john.doe password D60t7CBZ58k7JK2jxdlw4w== encrypted
```

Chapter 51 Location Service

Chapter contents

Introduction 608
Location Service configuration task list 608
Creating a Location Service 608
Adding a domain 608
Creating an identity 609
Authentication outbound face 610
Authentication inbound face 611
Registration outbound face 613
Registration inbound face 615
Call outbound face 616
Call inbound face 617
Creating an identity group 618
Inheriting from an identity group to an identity 618
Configuring the Message Waiting Indication feature for SIP 619
Subscription 619
Notification 620
Configuration 620
Message Waiting Indication through Call-Control 622
Configuration Examples 623

Introduction

This chapter describes how to configure location services in SmartWare.

Location Service configuration task list

The following section describes how to create a new location service and how to enter the configuration mode of an existing service. Additionally, it describes all commands and sub commands of the location service configuration mode. All configuration tasks for Location Services are listed below:

- Create a Location Service (see page 608)
- Add a domain (see page 608)
- Create an identity (see page 609)
- Create an identity group (see page 618)
- Inherit from an identity group to an identity (see page 618)

Creating a Location Service

The **location-service** command enters the configuration mode of a location service. If the requested service does not yet exists, a new one will be created. The no form of the command removes an existing location service.

Mode: Configure

Step	Command	Purpose
1	[*node*]**(cfg)# [no] location-service** *<name>*	Creates/Destroys a location service or enters configuration mode.

Adding a domain

The **domain** command specifies the domains that the location service is responsible for. If the application needs information from the location service, it performs a lookup with the Host Part of the Request-URI or the From-URI to find the right instance. The header selection from which the URI will be taken depends on the call direction (Outgoing/Incoming SIP-Call) and the requested information. The SIP environment determines which format the Domain has; it can either be a Domain-Name, a Host-Name or a Host-Address. If all components of the SIP environment are set up to operate in one specified domain, Domain is a Domain-Name. If point-to-point routing is used, Domain is either a Host-Name or a Host-Address. In this case, Host-Name is a FQDN (Full Qualified Domain Name).

Domain Examples:

Domain-Name: *biloxy.com*
Host-Name: *sip-ua.biloxy.com* or *sip-server.biloxy.com*
Host-Address: *192.168.10.1* or *10.10.10.1*

In case of point-to-point routing, local host-addresses may not be added to the domain list of a location service; the own addresses are known by the application. But, if an identity exists in two different location services

and the Context SIP Gateway has more than one transport binding, it is recommended to add the local host addresses as *Domain* to the appropriated location services.

Mode: Location Service

Step	Command	Purpose
1	[*node*]**(ls)[name]# [no] domain** *<name>* or [*node*]**(ls)[name]#domain** *<index> <name>* or [*node*]**(ls)[name]#domain before** *<index> <name>* or [*node*]**(ls)[name]#domain after** *<index> <name>*	Adds a new domain to the location service. If more than one domain has to be entered, the order of the list can be modified by using the index and/or the insert keywords *before* and *after*. The no form of the command removes an existing domain from the list.

Creating an identity

An identity represents one of multiple possible addresses over which a user is reachable (e.g. sip:john@inalp.com). Many configuration parameters related to call signaling, which were configured on the gateway or the sip interface, can now be configured per identity. This leads to a huge range of configuration possibilities in the identity.

According to the relationship between an identity and user, there can be many different aspects configured. If you are the user agent for a certain identity, use the outbound faces to specify the behavior when sending requests. If you are not the user agent of an identity but know this identity, then use the inbound faces to configure the behavior when this identity sends requests.

When creating an identity, it is important to consider that the name of the identity is always used as user-part when building a sip-uri. The name of the identity is also used when comparing to or matching with a sip-uri.

Mode: Location Service

Step	Command	Purpose
1	[*node*]**(ls)[name]# [no] identity** *<name>*	Adds a new identity to the location service. The no form of the command removes an existing identity.

Mode: Identity

Step	Command	Purpose
1	[*node*]**(identity)[name]# [no] display-name** *<display-name>*	Adds a display-name to the identity. The no form removes the display-name.

Mode: Identity

Step	Command	Purpose
1	[*node*]**(identity)[name]# [no] phone-context** *<phone-context>*	Adds a phone-context to the identity. The no form removes the phone-context.

An alias is an alternative way to express the user-part of an identity. An alias is never used to build a sip-uri and will never be used in communication with another device. The alias is used for comparing or matching the identity with a sip-uri received from an external device.

Mode: Identity

Step	Command	Purpose
1	[*node*]**(identity)[name]# [no] alias name** *<alias>*	Adds a new alias to the identity. The no form of the command removes an existing alias.

The huge amount of possible configuration parameters has been separated into different configuration entities–the faces. The faces refer to authentication, registration and call. Each face works in two directions–outbound and inbound. Outbound faces refer to to requests originating from your identity. Inbound faces refer to requests destined to your identity.

- Authentication outbound face (see page 610)
- Authentication inbound face (see page 611)
- Registration outbound face (see page 613)
- Registration inbound face (see page 615)
- Call outbound face (see page 616)
- Call inbound face (see page 617)

Authentication outbound face

The authentication outbound face is used to provide authentication credentials to challenges from other user agents or proxies.

Mode: Identity

Step	Command	Purpose
1	[*node*]**(identity)# [no] authentication outbound**	Adds a new face to the identity. The no form of the command removes an existing face with all content in it.

An authentication entry establishes a link between an identity and exactly one pair of credentials in an authentication-service. To link multiple credentials to an identity, there must be one authentication entry in the authentication outbound face for each pair of credentials to link. The parameter username selects the entry in the authentication-service to use. The parameter username can be omitted if and only if the name of the identity matches exactly the username in the authentication-service.

Mode: Authentication outbound

Step	Command	Purpose
1	[*node*]**(authout)#authenticate authentication-service** *<authentication-service>* **[username** *<username>***]** or [*node*]**(authout)#authenticate** *<index>* **authentication-service** *<authentication-service>* **[username** *<username>***]** or [*node*]**(authout)#authenticate before** *<index>* **authentication-service** *<authentication-service>* **[username** *<username>***]** or [*node*]**(authout)#authenticate after** *<index>* **authentication-service** *<authentication-service>* **[username** *<username>***]**	Adds a new authentication entry to the authentication outbound face. If more than one authentication entry has to be entered, the order of the list can be modified by using the index and/or the insert keywords before and after.

Mode: Authentication outbound

Step	Command	Purpose
1	[*node*]**(authout)#no authenticate [***<index>***]**	Removes the authentication entry at the index or removes all authentication entries if no index is given.

Mode: Authentication outbound

Step	Command	Purpose
1	[*node*]**(authout)#authenticate none**	Removes all authentication entries and disables explicitly authentication outbound.

Authentication inbound face

The authentication inbound face is used when you want to challenge other user agents.

Mode: Identity

Step	Command	Purpose
1	[*node*]**(identity)[name]# [no] authentication inbound**	Adds a new face to the identity. The no form of the command removes an existing face with all content in it.

An authentication entry establishes a link between an identity and a pair of credentials in an authentication-service. To link multiple credentials to an identity, there must be one authentication entry in the authentication inbound face for each pair of credentials to link. The parameter username selects the entry in the authentication-service to use. The parameter username can be omitted if and only if the name of the identity matches the username in the authentication-service exactly.

Mode: Authentication inbound

Step	Command	Purpose
1	[*node*]**(authin)#authenticate authentication-service** *<authentication-service>* **[username** *<username>***]** or [*node*]**(authin)#authenticate** *<index>* **authentication-service** *<authentication-service>* **[username** *<username>***]** or [*node*]**(authin)#authenticate before** *<index>* **authentication-service** *<authentication-service>* **[username** *<username>***]** or [*node*]**(authin)#authenticate after** *<index>* **authentication-service** *<authentication-service>* **[username** *<username>***]**	Adds a new authentication entry to the authentication inbound face. If more than one authentication entry has to be entered, the order of the list can be modified by using the index and/or the insert keywords before and after.

Mode: Authentication inbound

Step	Command	Purpose
1	[*node*]**(authin)#no authenticate [***<index>***]**	Removes the authentication entry at the index or removes all authentication entries if no index is given.

Mode: Authentication inbound

Step	Command	Purpose
1	[*node*]**(authin)#authenticate none**	Removes all authentication entries and disables explicitly authentication inbound.

Registration outbound face

The registration outbound face is used to register an identity on an external registrar. Then, the registrar forwards calls from the registered identity to your identity.

Mode: Identity

Step	Command	Purpose
1	[*node*]**(identity)[name]# [no] registration outbound**	Adds a new face to the identity. The no form of the command removes an existing face with all content in it.

Mode: Registration outbound

Step	Command	Purpose
1	[*node*]**(regout)# [no] register [auto|none]**	Enables registration with auto or disables registration explicitly with none.

Mode: Registration outbound

Step	Command	Purpose
1	[*node*]**(regout)# [no] registrar** *<host>* [*<port>*]	Configures the address of the registrar to send your register requests. When no registrar is configured, the register requests are sent to the first domain entry in the location-service.

Mode: Registration outbound

Step	Command	Purpose
1	[*node*]**(regout)# [no] lifetime** *<seconds>*	Configures the desired lifetime of the registration. When no lifetime is configured the desired lifetime is set to 3600 seconds.

Mode: Registration outbound

Step	Command	Purpose
1	[*node*]**(regout)# [no] retry-timeout (on-client-error|on-sysstem-error|on-server-error)** *<seconds>*	Configures the time to wait after a failed registration according to three different error categories. After this time we begin to register from new. If no retry-timeout is configured the retry-timeout is set to 10 seconds.

Mode: Registration outbound

Step	Command	Purpose
1	[*node*]**(regout)#proxy** *<host>* **[***<port>***] [strict-route]** or [*node*]**(regout)#proxy** *<index> <host>* **[***<port>***] [strict-route]** or [*node*]**(regout)#proxy before** *<index> <host>* **[***<port>***] [strict-route]** or [*node*]**(regout)#proxy after** *<index> <host>* **[***<port>***] [strict-route]**	Adds a new proxy entry to the registration outbound face. For each proxy configured there is a route-header added. If more than one proxy entry has to be entered, the order of the list can be modified by using the index and/or the insert keywords before and after.

Mode: Registration outbound

Step	Command	Purpose
1	[*node*]**(regout)#proxy** *<index>* **down** *<positions>* or [*node*]**(regout)#proxy** *<index>* **up** *<positions>*	If multiple proxies are configured the entry at the index can be moved in the proxy list up or down the number of positions given in the command.

Mode: Registration outbound

Step	Command	Purpose
1	[*node*]**(regout)#no proxy [***<index>***]**	Removes the proxy entry at the index or remove all proxy entries if no index is given.

Mode: Registration outbound

Step	Command	Purpose
1	[*node*]**(regout)#proxy none**	Removes all proxy entries and disables explicitly the use of a proxy.

Registration inbound face

The registration inbound face is used when you want to allow external user agents to register, so that you can route calls to the registered contacts.

Mode: Identity

Step	Command	Purpose
1	[*node*]**(identity)[name]# [no] registration inbound**	Adds a new face to the identity. The no form of the command removes an existing face with all content in it.

Mode: Registration inbound

Step	Command	Purpose
1	[*node*]**(regin)# [no] lifetime [default** *<seconds>*] **[min** *<seconds>*] **[max** *<seconds>*]	Configures the range of the expiration time accepted for inbound registration. If there is a time requested which is out of the range the time is set to a value which fits the range. If there is no time requested it is set to the default lifetime.

Mode: Registration inbound

Step	Command	Purpose
1	[*node*]**(regin)# [no] contact** *<host>* **[***<port>***]** *<context cs> <interface sip>* **[priority** *<prom-ille>*	Adds a contact to the registration inbound face. The interface and context parameter defines where the sip-location-service routes a call destined to that contact. The higher priority a contact has the sooner a contact is chosen. The default priority is 1000 which is also the maximum priority. The no form of the command removes a contact.

Call outbound face

The call outbound face is used to configure call properties for outgoing calls.

Mode: Identity

Step	Command	Purpose
1	[*node*]**(identity)[name]# [no] call outbound**	Adds a new face to the identity. The no form of the command removes an existing face with all content in it.

Mode: Call outbound

Step	Command	Purpose
1	[*node*]**(callout)[name]# [no] use profile (sip|tone-set|voip)** *<profile>*	Adds or removes a profile to use if an outgoing call destines this identity

Mode: Call outbound

Step	Command	Purpose
1	[*node*]**(callout)[name]# [no] preferred-transport-protocol (tcp|udp)**	Selects which protocol to prefer if an outgoing call destines this identity.

Mode: Call outbound

Step	Command	Purpose
1	[*node*]**(callout)[name]# [no] traffic-class** *<traffic-class>*	Selects which traffic class to set if an outgoing call destines this identity.

Mode: Call outbound

Step	Command	Purpose
1	[*node*]**(callout)#proxy** *<host>* **[***<port>***] [strict-route]** or [*node*]**(callout)#proxy** *<index> <host>* **[***<port>***] [strict-route]** or [*node*]**(callout)#proxy before** *<index> <host>* **[***<port>***] [strict-route]** or [*node*]**(callout)#proxy after** *<index> <host>* **[***<port>***] [strict-route]**	Adds a new proxy entry to the call outbound face. If the from-uri of a call originating from us matches the identity for each proxy configured there is a route-header added. If more than one proxy entry has to be entered, the order of the list can be modified by using the index and/or the insert keywords before and after.

Mode: Call outbound

Step	Command	Purpose
1	[*node*]**(callout)#proxy** *<index>* **down** *<positions>* or [*node*]**(callout)#proxy** *<index>* **up** *<positions>*	If multiple proxies are configured the entry at the index can be moved in the proxy list up or down the number of positions given in the command.

Mode: Call outbound

Step	Command	Purpose
1	[*node*]**(callout)#no proxy** [*<index>*]	Removes the proxy entry at the index or remove all proxy entries if no index is given.

Mode: Call outbound

Step	Command	Purpose
1	[*node*]**(callout)#proxy none**	Removes all proxy entries and disables explicitly the use of a proxy.

Call inbound face

The call inbound face is used to configure call properties for incoming calls.

Mode: Identity

Step	Command	Purpose
1	[*node*]**(identity)[name]# [no] call inbound**	Adds a new face to the identity. The no form of the command removes an existing face with all content in it.

Mode: Call inbound

Step	Command	Purpose
1	[*node*]**(callin)[name]# [no] use profile (sip|tone-set|voip)** *<profile>*	Adds or removes a profile to use if an incoming call destines this identity.

Creating an identity group

Multiple identities with the same properties can be grouped in an identity-group. The identity-group can be configured exactly in the same way and with the same parameters as an identity. An identity-group can only inherit configurations to identities, but they never play an active role.

The special identity-group "default" inherits parameters to identities which are unknown or not configured. Even dynamically created identities from registration inbound inherits from the identity-group "default". Configured identities do not inherit from the identity-group "default" unless is it explicitly configured to do so.

Mode: Location Service

Step	Command	Purpose
1	[*node*](ls)[name]# [no] identity-group *<name>*	Adds a new identity-group to the location service. The no form of the command removes an existing identity-group.

Inheriting from an identity group to an identity

An identity only inherits parameters from an identity-group if the parameters are not configured in the identity itself. Some commands allow the identity to explicitly disable some configurations that were otherwise inherited.

Mode: Location Service

Step	Command	Purpose
1	[*node*](ls)[name]#identity *<name>* inheritance *<identity-group>*	Configures the identity to inherit parameters which are not configured from the identity-group.

Mode: Location Service

Step	Command	Purpose
1	[*node*](ls)[name]#identity-group *<name>* inheritance *<identity-group>*	Configures the identity-group to inherit parameters which are not configured from another identity-group.

Mode: Location Service

Step	Command	Purpose
1	[*node*](ls)[name]#identity *<name>* no-inheritance	Configures the identity to not inherit parameters from the identity-group.

Mode: Location Service

Step	Command	Purpose
1	[*node*](ls)[name]#identity-group *<name>* no-inheritance	Configures the identity-group to not inherit parameters from the other identity-group.

Configuring the Message Waiting Indication feature for SIP

Note Message Waiting Indication is programmed in two sections of SmartWare, the FXS interface, and the SIP Location service. The information below refers to information for configuring the Message Waiting Indication feature for SIP. For information on configuring the Message Waiting Indication feature for FXS, see "Configuring the Message Waiting Indication feature for FXS" on page 408.

The SIP part of SmartWare is now able to receive message waiting information. This is done according to the subscribe notify mechanism described in RFC 3265 and RFC 3842. To be compatible with current installations, it is possible to choose between *explicit* and *implicit subscription*. In **explicit subscription**, SmartWare establishes the subscription by sending SUBSCRIBE messages to a configured message server. In **implicit subscription**, the message server gets the subscriber information due to configuration or another mechanism like registration. In any case, SmartWare accepts and processes NOTIFY messages with message waiting information, which is then forwarded to the according destination.

Subscription

If an identity is added to a location-service which is bound to a context sip-gateway, the following procedure takes place:

1. Determine if identity should be subscribed.
   a. The location-service containing the identity must have at least one domain configured.
   b. The identity must have a message inbound face configured or inherited.
   c. The subscription command must be set to "explicit".
2. Build the address to subscribe. The name of the identity builds the user-part. The first domain configured in the location-service builds the host-part.
3. Build the address of the message server. The message server configured in the message inbound face is taken as request-uri. If no message server is configured the first domain configured in the location-service builds the request-uri.
4. Build expire header. If a lifetime is configured in the message inbound face the expire header is set to desired lifetime else the lifetime is set to 3600 seconds.
5. Build contact address. The spoofed-contact parameter of the sip-interface in the context sip gateway, through which the SUBSCRIBE request is sent, is set as contact address. If no spoofed-contact is configured the ip-address and port of the sip-interface in the context sip-gateway through which the SUBSCRIBE request is sent builds the contact address.
6. Send the SUBSCRIBE request.

If one of these steps has no result and fails, the subscription fails. After a certain timeout (which is configurable in the registration outbound face), the request is re-issued.

Outgoing SUBSCRIBE requests can provide authentication credentials. See "Outbound Authentication" on page 561 for more information.

Notification

If the gateway receives an incoming NOTIFY request, the following procedure takes place:

1. Determine to which sip interface in the context cs the request should be forwarded. This happens according the same rules as an incoming INVITE is forwarded.
2. Get Identity. All location services bound to the context sip-gateway are searched for the identity:
   a. the identity matching the to-uri
   b. the identity matching request-uri
   c. the identity-group default
3. Get message inbound face. The identity found must have a message inbound face configured.
4. Check subscription. The option “subscription” must be set to “implicit” or “explicit”. In the case of explicit subscription the real state of the subscription is not examined. According to RFC 3265 NOTIFY requests must be handled even before subscription is completed.
5. Check event header. The event-header of the NOTIFY request must be set to “message-summary”.
6. Check content header. If present, the content header must be set to “application/simple-message-summary”
7. Forward message waiting information. Forward content of NOTIFY message through call-control according normal call routing to the destination provider.
8. Return 200 OK if all of these steps are successful.

If one of these steps fails the notification fails and an according message is sent.

Incoming NOTIFY requests can be challenged to provide authentication credentials. “Inbound Authentication” on page 562 for more information.

Configuration

For configuration purposes, there is a new configuration face created in the identity of the location service. The message inbound face is used to subscribe an identity on a message server and enables the reception of NOTIFY messages with message waiting information.

Mode: Identity

Step	Command	Purpose
1	[*node*]**(identity)[name]#[no] message inbound**	Adds a new face to the identity. The no form of the command removes an existing face with all content in it.

Mode: Message inbound

Step	Command	Purpose
1	[*node*]**(msgin)#[no] subscribe [implicit | explicit | none]**	Enables subscription implicit, explicit or disables subscription with none.

Mode: Message inbound

Step	Command	Purpose
1	[*node*]**(msgin)#[no] message server** *<host>* **[***<port>***]**	Configures the address of the message server to send your subscription requests for explicit subscription. When no message-server is configured, the subscription requests are sent to the first domain entry in the location-service.

Mode: Message inbound

Step	Command	Purpose
1	[*node*]**(msgin)#[no] lifetime** *<seconds>*	Configures the desired lifetime of the explicit subscription. When no lifetime is configured the desired lifetime is set to 3600 seconds.

Mode: Message inbound

Step	Command	Purpose
1	[*node*]**(msgin)#[no] retry-timeout (on-client-error | on-system-error | on-server-error]** *<seconds>*	Configures the time to wait after a failed subscription according to three different error categories. After this time we begin to subscribe from new. If no retry-timeout is configured the retry-timeout is set to 10 seconds.

Mode: Message inbound

Step	Command	Purpose
1	[*node*]**(msgin)#proxy** *<host>* **[***<port>***] [strict-route]** *or* [*node*]**(msgin)#proxy** *<index> <host>* **[***<port>***] [strict-route]** *or* [*node*]**(msgin)#proxy before** *<index> <host>* **[***<port>***] [strict-route]** *or* [*node*]**(msgin)#proxy after** *<index> <host>* **[***<port>***] [strict-route]**	Adds a new proxy entry to the message inbound face. For each proxy configured there is a route-header added to the SUBSCRIBE requests in explicit subscription. If more than one proxy entry has to be entered, the order of the list can be modified by using the index and/or the insert keywords before and after.

Mode: Message inbound

Step	Command	Purpose
1	[*node*]**(msgin)#proxy none**	Removes all proxy entries and disables explicitly the use of a proxy.

Message Waiting Indication through Call-Control

The message waiting indication received on the SIP side is transported through the call control to the destination FXS interface. Currently, the SIP interface is the only source of message waiting indication information and the FXS interface is the only possible destination of this information. The information is routed like a call through the call-control. Therefore, it must exist a valid routing path for a call from the SIP interface handling the NOTIFY to the destination fxs interface. This must be the case, because the phone connected to the FXS interface must be able to get calls from SIP before it makes any sense to get message waiting information for missed calls from SIP.

The message waiting information can be transported through routing-tables, service aaa, service limiter, service priority, service distribution-group and service hunt-group.

- In case of routing through **routing tables**, the message waiting information will reach the right destination only if the parameter according which the routing occurs is set to exactly the same value in an incoming INVITE request for that FXS interface as in the incoming NOTIFY request for that FXS interface.
- In the case of "**called-e164**", this is surely the same, but pay attention to the manipulation of parameters.
- In case of service **distribution-group**, the message waiting information is transported to any valid destination of the distribution-group.
- In case of service **hunt-group**, the message waiting information is transported only to the first destination of the hunt group.
- In **any other of the named services**, the message waiting information is routed transparently through.

If the message waiting information is routed to a destination service or interface which is not explicitly listed here, the information will be dropped.

Configuration Examples

In this configuration example, inheritance is used.

Example:

```
location-service INALP
  domain inalp.com

  identity-group REGISTER

    authentication outbound
      authenticate 1 authentication-service AUTH_INALP username john

    registration outbound
      registrar sip.inalp.com
      lifetime 600
      register auto

  identity 300 inherits REGISTER
  identity
  identity 400 inherits REGISTER
```

Exactly the same can be configured without inheritance. All inherited parameters can be configured in the identity itself. Inheritance is useful if multiple identities share the same configuration.

Example:

```
location-service INALP
  domain inalp.com

  identity 300

    authentication outbound
      authenticate 1 authentication-service AUTH_INALP username john

    registration outbound
      registrar sip.inalp.com
      lifetime 600
      register auto

  identity 400

    authentication outbound
      authenticate 1 authentication-service AUTH_INALP username john

    registration outbound
      registrar sip.inalp.com
      lifetime 600
      register auto
```

Chapter 52 VoIP debugging

Chapter contents

Introduction 625
Debugging strategy 625
Filtering debug monitor output 626
Verifying IP connectivity 626
Debugging call signaling 627
Debugging ISDN signaling 627
Verify an incoming call 628
Verify an outgoing call 629
Verify ISDN layer 2 and 3 status 631
Debugging FXS Signaling 632
Verify an incoming call 632
Verify an outgoing call 633
Debugging H.323 Signaling 634
Verify an incoming call 634
Verify an outgoing call 636
Debugging SIP signaling 638
Verify an incoming call 638
Verify an outgoing call 638
Using SmartWare's internal call generator 639
Debugging voice data 640
Check system logs 642
How to submit trouble reports to Patton 642

Introduction

This chapter describes how to debug VoIP sessions, including the signaling part and the voice data path part (speech, fax, and modem connectivity). It provides debugging strategies to help locate the source of a problem, and describes the **show** and **debug** commands used to verify correct system operation and to troubleshoot problems.

This chapter includes the following sections:

- Debugging strategy
- Verifying IP connectivity
- Debugging call signaling (see page 627)
- Debugging voice data (see page 640)

Debugging strategy

Multi-service IP networks comprise highly sophisticated systems and protocols that offer a great many possibilities. Unfortunately, the possible sources of trouble are almost as many, so it is important to use a very methodical approach when tracking down a problem:

- Work from the bottom to the top of the protocol stack. Verify that cables and connectors are in good shape, verify the link layer, and check IP connectivity before working on application problems.
- Work from the core to the edge. Problems always show up end-to-end, the phone does not ring, or the browser cannot find the web site. To track down network problems it is however helpful to start with a minimal number of hops, make sure everything is ok and then increase the end-to-end distance hop by hop.

Enabling some or all debug monitors may degrade system performance (IP routing, call signaling). To avoid inadvertent permanent system performance degradation make sure all monitors are switched off once the configuration is debugged and running.

Note Event log files record warnings and other information from system components. Entries in the logs are time-stamped with the actual system time, so make sure the SmartNode always has the actual time as its system time. Otherwise, you are not be able to get some cleverly information from the event logs because the time stamps are always unusable.

You can enter the system time manually or have it be automatically set via SNTP. Refer to chapter 7, "Basic system management" on page 90, or chapter 26, "SNTP client configuration" on page 282.

Filtering debug monitor output

The output of the debug monitors can be filtered using the following command to let the terminal only print important information. The specified expression is a regular expression, which is used by the filter to select important lines.

Mode: Configure

Step	Command	Purpose
1	[*name*]**(cfg)# [no] terminal monitor-filter** *<expression>*	Enables the monitor filter using the specified expression.

The following example activates a filter, which causes the debug monitors to print only lines containing the word *ERROR* on the terminal:

```
terminal monitor-filter .*ERROR
```

Note Only one filter expression can be active at a time. If you enter the command a second time, the new filter expression will replace the old one.

The following command will disable the filter completely:

```
no terminal monitor-filter
```

Verifying IP connectivity

This procedure describes how to use the **ping** command to test IP connectivity. It verifies that your SmartNode can communicate with such hosts as a gatekeeper, and IP phone, a registrar, and other VoIP gateways.

Use Telnet to access your SmartNode, then use the **ping** command to verify that an IP packet can be sent to, and received from, all hosts with which the SmartNode should be able to communicate (e.g. Gatekeeper, IP phone, Registrar, other VoIP gateways).

Mode: Administrator execution

Step	Command	Purpose
1	***unit*#ping** *ipaddress* [*number-of-packets*] [**timeout** *seconds*]	Determines whether an IP host can be contacted.

Example: Verify IP connectivity

The following example uses the **ping** command to test connections to remote hosts 192.195.23.10 and 172.16.40.122. The results show an unsuccessful attempt to contact host 192.195.23.10 (as indicated by the “No

route to host" response) and a successful connection to host 172.16.40.122 (which received and sent packets with no loss).

```
node#ping 192.195.23.1 10 timeout 5
Sending 10 ICMP echo requests to 192.195.23.10, timeout is 5 seconds:
% No route to host
node#
node#ping 172.16.40.122
Sending 5 ICMP echo requests to 172.16.40.122, timeout is 1 seconds:
Reply from 172.16.40.122: Time <10ms
Reply from 172.16.40.122: Time <10ms
Reply from 172.16.40.122: Time <10ms
Reply from 172.16.40.122: Time <10ms
Reply from 172.16.40.122: Time <10ms
Ping statistics for 172.16.40.122:
  Packets: Sent 5, Received 5, Lost 0 (0% loss),
  RTT:     Minimum <10ms, Maximum <10ms, Average <10ms
```

Debugging call signaling

If calls do not connect, or disconnect during the process of connecting, there is a problem in call signaling. We suggest the following steps to debug call signaling:

1. Work from the call source to the call destination.
2. Make sure that the call enters correctly the context CS of your unit (debug the source signaling protocol, depending on where the call comes from).
3. Make sure that the call leaves correctly the context CS of your unit (debug the destination signaling protocol, depending on where the call goes to).
4. Debug call routing when the call enters the context CS, but it does not leave it. Remember that context CS must be activated ("no shutdown") for call routing to work.

Please make yourself familiar with the context CS concept described in chapter 31, "CS context overview" on page 339. The following terminology will be used in this chapter:

- *Incoming call*: Call setup attempt from a call signaling protocol towards the context CS
- *Outgoing call*: Call setup attempt from the context CS towards a call signaling protocol

A basic call from an ISDN terminal connected to your unit over SIP to another SIP gateway is thus an *incoming* and *outgoing* call a the same time, from the context CS perspective: It *comes in* from ISDN and *goes out* over SIP.

Debugging ISDN signaling

Overview: ISDN debug monitors

	Command	Purpose
	***unit*#debug ccisdn signaling**	Prints all ISDN layer 3 signaling messages, and call control related activity of the ISDN interface. This is a good monitor to start with when debugging ISDN.

	Command	Purpose
	***unit*#debug ccisdn error**	Prints all errors occurring in ISDN call control and ISDN datapath control. Always switch this monitor on when debugging ISDN.
	***unit*#debug ccisdn datapath**	Prints operations on the ISDN part of the voice data path. Use this monitor if you experience problems in the data path (no speech connectivity, speech only in one direction). This monitor is not needed to debug signaling.
	***unit*#debug isdn error**	Prints all errors occurring on the ISDN port (protocol stack layers 1 to 3). Always switch this monitor on when debugging ISDN.
	***unit*#debug isdn event** *slot port* **{all | layer2 | layer3}**	Logs in detail the operation on the ISDN port (protocol stack layers 2 to 3).
	***unit*#show port isdn** *slot port* **status**	Shows the status of an ISDN port. Among others, indicates the link state of that port.
	unit*#show call-control provider** *name* **[detail** *detail]**	Shows the status of an ISDN interface in context CS (call control part of ISDN signaling). Shown are all calls currently ongoing on the interface, with their call states, signaling peers and voice data path parameters. *name* is the name of the ISDN interface. Use *detail* 5 for most verbose output.

Verify an incoming call

Make sure an incoming call is entering correctly context CS. In this example, an ISDN terminal connected to an ISDN port places a call. The port is in NET mode. The port is bound to a context CS interface named *TERMINAL*. The debug output below shows a normal (working) call setup sequence.

```
unit>enable
unit#configure
unit(cfg)#debug ccisdn error
unit(cfg)#debug ccisdn signaling
unit(cfg)#debug isdn error
unit(cfg)#18:34:10  ICC   > [TERMINAL] << Message: primitive=64
18:34:10  ICC   > [TERMINAL] Added endpoint TERMINAL-00b73348
18:34:10  ICC   > [TERMINAL] NEW CALL. Allocated Endpoint TERMINAL-00b73348
18:34:10  ICC   > [TERMINAL-00b73348] << SETUP (DSS1 Ntwk)
  Bearer capability : speech - CCITT
    circuit mode - 64kBit/s - G.711 A-law
  Calling party number : 60
    unknown number - unknown numbering plan
    presentation allowed - user provided not screened
  Called party number : 50
    unknown number - E.164 numbering plan
  High layer compatibility : telephony
    CCITT

18:34:10  ICC   > [TERMINAL-00b73348] State: NULL, Event: TERMINAL SETUP IND
18:34:10  ICC   > [TERMINAL-00b73348] Set state to OVERLAP SENDING
```

```
18:34:10  ICC   > [TERMINAL-00b73348] >> SETUP ACKNOWLEDGEMENT (DSS1 Ntwk)

18:34:10  ICC   > [TERMINAL-00b73348] State: OVERLAP SENDING, Event: PEER TRYING
18:34:10  ICC   > [TERMINAL-00b73348] State: OVERLAP SENDING, Event: PEER ALERTING
18:34:10  ICC   > [TERMINAL-00b73348] Set state to CALL DELIVERED
18:34:10  ICC   > [TERMINAL-00b73348] >> ALERTING (DSS1 Ntwk)
  Progress indicator : inband information available
    private network serving local user - CCITT

18:34:18  ICC   > [TERMINAL-00b73348] State: CALL DELIVERED, Event: PEER CONNECTED
18:34:18  ICC   > [TERMINAL-00b73348] Set state to ACTIVE
18:34:18  ICC   > [TERMINAL-00b73348] >> CONNECT (DSS1 Ntwk)
  Connected number : 50
    unknown number - E.164 numbering plan
    presentation allowed - user provided not screened
```

Explanation:

- The terminal places the call using en-bloc sending (no overlap dialing). In the log, this is represented by the lines **`18:34:10 ICC > [TERMINAL-00b73348] << SETUP (DSS1 Ntwk)`** and below. This means there is a message coming from ISDN as represented by the left angle brackets (<<). ISDN layer1, 2 and 3 work correctly in this case.
- If the *SETUP* message does not appear, one of the lower ISDN layers doesn't work. Verify ISDN port status using the **show port isdn** command , and the "debug isdn events" commands (see below).
- The *SETUP* message contains different elements, among others the calling party number (60), and the called party number (50). Verify these (depending on your application, there might be as well other elements in the message).
- The line **`18:34:10 ICC > [TERMINAL-00b73348] >> ALERTING (DSS1 Ntwk)`** shows that the dialed number is alerting now. The *ALERTING* message is sent back to the terminals represented by the right angle brackets (>>). You can be sure now that context CS functionality is working.
- The line **`18:34:18 ICC > [TERMINAL-00b73348] >> CONNECT (DSS1 Ntwk)`** indicates that the call is now established. The called party has answered the call.
- If instead of the *ALERTING*, a *RELEASE* or *DISCONNECT* message appears, continue debugging the outgoing call on the destination signaling protocol and the call-router.

Verify an outgoing call

Make sure a call from context CS is accepted by the connected ISDN terminal or the PSTN. In this example, an ISDN port is connected to the PSTN. The port is bound to a context CS interface named PSTN.

The simplest way to verify the signaling of an outgoing call is to use the built-in call generator. You can dial any number you know is reachable over this ISDN line. If it is the PSTN, you can dial e.g. your mobile phone number.

```
unit>enable
unit#configure
unit(cfg)#debug ccisdn error
unit(cfg)#debug ccisdn signaling
unit(cfg)#debug isdn error
unit(cfg)#call 123456 dial 987654321 dest-interface PSTN
unit(cfg)#22:03:06  ICC   > [PSTN] Added endpoint PSTN-00b70a20
```

```
22:03:06  ICC   > [PSTN] NEW CALL. Allocated Endpoint PSTN-00b70a20
22:03:06  ICC   > [PSTN-00b70a20] >> SETUP (DSS1 User)
  Bearer capability : speech - CCITT
    circuit mode - 64kBit/s - G.711 A-law
  Calling party number : 123456
    unknown number - unknown numbering plan
    presentation allowed - user provided not screened
  Called party number : 987654321
    unknown number - unknown numbering plan
  High layer compatibility : telephony
    CCITT

22:03:06  ICC   > [PSTN-00b70a20] Set state to CALL PRESENT
22:03:06  ICC   > [PSTN-00b70a20] State: CALL PRESENT, Event: PEER CONNECTED
22:03:06  ICC   > [PSTN] << Message: primitive=31
22:03:06  ICC   > [PSTN-00b70a20] << ALERTING (DSS1 User)

22:03:06  ICC   > [PSTN-00b70a20] State: CALL PRESENT, Event: PSTN ALERTING IND
22:03:06  ICC   > [PSTN-00b70a20] Set state to CALL RECEIVED

unit(cfg)#call 123456 drop
unit(cfg)#22:03:14  ICC   > [PSTN-00b70a20] State: CALL RECEIVED, Event: PEER
RELEASED
22:03:14  ICC   > [PSTN-00b70a20] Set state to DISCONNECT INDICATION
22:03:14  ICC   > [PSTN-00b70a20] >> DISCONNECT (DSS1 User)
  Cause : normal call clearing
    private network serving local user - CCITT - Q.931
  Progress indicator : inband information available
    private network serving local user - CCITT

22:03:14  ICC   > [PSTN] << Message: primitive=50
22:03:14  ICC   > [PSTN-00b70a20] << RELEASE COMPLETE (DSS1 User)

22:03:14  ICC   > [PSTN-00b70a20] State: DISCONNECT INDICATION, Event: PSTN RELEASE
IND
22:03:14  ICC   > [PSTN-00b70a20] Set state to NULL
22:03:14  ICC   > [PSTN] CLEARING CALL PSTN-00b70a20
22:03:14  ICC   > [PSTN] Removed endpoint PSTN-00b70a20
22:03:14  ICC   > [PSTN] Destroying finished calls.
22:03:14  ICC   > [PSTN] Destroyed endpoint PSTN-00b70a20
```

Explanation:

- Place a call on the command line using the line **`unit(cfg)##call 123456 dial 987654321 dest-interface PSTN`**. The calling party number is *123456* and the called party number *987654321*. Replace the called party with your mobile phone number, or any other number you know is reachable over the interface.
- Verify the called party number in the *SETUP* message on the consecutive lines.
- The line **`22:03:06  ICC   > [PSTN-00b70a20] << ALERTING (DSS1 User)`** indicates that the called party is ringing now, and has sent back the *ALERTING* message to us. This means that the ISDN layer 1, 2, 3 and that the ISDN signaling works.
- If there is a *RELEASE* or *DISCONNECT* message instead of the *ALERTING*, either ISDN connectivity is not working (i.e. the message cannot be sent to the ISDN line), or the PSTN has rejected the call. Verify

ISDN port status using the **show port isdn** command, and the **debug isdn events** commands (see below).

- Because the *ALERTING* is indication enough that the signaling is working, we drop the call from the command line: **`unit(cfg)#call 123456 drop`**
- The interface then sends a *DISCONNECT* message to the line, and the PSTN answers with a *RELEASE COMPLETE*

Verify ISDN layer 2 and 3 status

ISDN layer 2 and 3 can be verified using a **show** command:

```
node(cfg)#show port isdn 0 2 status
Logical Isdn Driver: 0 0
========================

  Slot:                               0
  Number of Ports:                    5

  Statistics
  ----------

    Leased buffers:                   24
    Max. leased buffers:              34
    Next Call Key:                    0

  Logical Port: 0 0 2
  -------------------

    Admin State:                      Open
    Real State:                       Open

    Operating Layer:                  3
      Link State:                     up

    Layer 2
      Permanent Layer 2:              off
      Protocol:                       PointToMultiPoint
      UniSide:                        Net

    Layer 3
      Protocol:                       Dss1
      UniSide:                        Net
      MinChannel:                     0
      MaxChannel:                     1
      MaxCalls:                       2
      Hunt Mode:                      Ascending
      Signalling Mode:                Etsi
```

The line “link state: up” tells you that layer 1 is up. To debug the layers 2-3 state machines, use the command **debug isdn event** *slot port* **{all | layer2 | layer3}.**

Debugging FXS Signaling

Overview: FXS debug monitors

	Command	Purpose
	***unit*#debug ccfxs**	Prints all operations on the FXS interfaces (high-level).
	***unit*#debug fxs**	Prints all operations on the FXS ports (low-level).
	***unit*#debug media-gateway control**	Prints the dialed digits.
	unit*#show call-control provider** *name* **[detail** *detail]**	Shows the status of an FXS interface in context CS (call control part of FXS signaling). Shown are all calls currently ongoing on the interface, with their call states, signaling peers and voice data path parameters. *name* is the name of the FXS interface. Use *detail 5* for most verbose output.

Verify an incoming call

Make sure an incoming call is entering correctly context CS. In this example, a POTS terminal connected to an FXS port goes off hook and places a call. The port is bound to a context CS interface named PHONE. The debug output below shows a normal (working) call setup sequence.

```
unit>enable
unit#configure
unit(cfg)#debug ccfxs
unit(cfg)#debug fxs
unit(cfg)#09:00:11  FXS   > [0 1] Off hook.
09:00:11  FXS   > [0 1] Notifying off hook.
09:00:11  FXS   > FXS-00/01: state=on-hook, event=off-hook
09:00:11  FXS   > [0 1] Set state to 'Active'
09:00:11  FXS   > FXS-00/01: new state=filtering-events
09:00:11  CFXS  > [EP PHONE] Change state to DIAL-TONE.
09:00:11  CFXS  > [EP PHONE] Change datapath direction to receive-only.
09:00:11  CFXS  > [EP PHONE] Play tone: dial-tone
09:00:11  FXS   > FXS-00/01: state=filtering-events, event=timeout
09:00:11  FXS   > FXS-00/01: new state=off-hook
09:00:13  CFXS  > [EP PHONE] Change state to DIALING.
09:00:13  CFXS  > [EP PHONE] Stop tone.
09:00:43  CFXS  > [EP PHONE] Play tone: ringback-tone
09:01:01  CFXS  > [EP PHONE] Change state to CONNECTED.
09:01:01  CFXS  > [EP PHONE] Stop tone.
09:01:01  CFXS  > [EP PHONE] Change datapath direction to send/receive.
```

Explanation:

- **`unit(cfg)#09:00:11  FXS > [0 1] Off hook.`**: The phone went off-hook
- **`09:00:11  CFXS > [EP PHONE] Change state to DIAL-TONE.`**: The dial-tone is played.
- **`09:00:13  CFXS > [EP PHONE] Change state to DIALING.`**: The first digit has been touched. Dial-tone stops.

- **09:00:43 CFXS > [EP PHONE] Play tone: ringback-tone**: The called-party is ringing, alerting-tone is played to the phone. This means that the call arrived correctly in context CS, and has reached its destination.
- If the above line does not appear, the call has arrived in context CS, but has not reached its destination. Continue debugging call-router and the outgoing call to the destination signaling protocol.
- If you want to see the dialed digits in the debug monitor, use “debug media-gateway control”.
- **09:01:01 CFXS > [EP PHONE] Change state to CONNECTED.**: the called party has accepted the call, and the call is now established.

Verify an outgoing call

Make sure a call from context CS is accepted by the connected POTS terminal. The port is bound to a context CS interface named PHONE.

The simplest way to verify the signaling of an outgoing call is to use the built-in call generator. You can place a call to any called-party, and verify that the connected phone is ringing.

```
unit>enable
unit#configure
unit(cfg)#debug ccfxs
unit(cfg)#debug fxs
unit(cfg)#call 1 dial 2 dest-interface PHONE
unit(cfg)#09:13:55  CFXS  > [EP PHONE] Change state to RINGING.
09:13:55  CFXS  > [EP PHONE] Start Ring.
09:13:55  FXS   > FXS-00/01: state=on-hook, event=ring-start
09:13:55  FXS   > [0 1] Set state to 'Ringing'
09:13:55  FXS   > FXS-00/01: new state=ringing
09:13:56  FXS   > [0 1] Set state to 'RingPause'
09:14:00  FXS   > [0 1] Set state to 'Ringing'
09:14:01  FXS   > [0 1] Set state to 'RingPause'
09:14:02  FXS   > [0 1] Off hook.
09:14:02  FXS   > [0 1] Notifying off hook.
09:14:02  FXS   > FXS-00/01: state=ringing, event=off-hook
09:14:02  FXS   > [0 1] Set state to 'Idle'
09:14:02  FXS   > [0 1] Set state to 'Active'
09:14:02  FXS   > FXS-00/01: new state=off-hook
09:14:02  CFXS  > [EP PHONE] Change state to CONNECTED.
09:14:02  CFXS  > [EP PHONE] Stop ring.
09:14:02  FXS   > FXS-00/01: state=off-hook, event=ring-stop
09:14:02  FXS   > FXS-00/01: new state=off-hook
09:14:02  CFXS  > [EP PHONE] Change datapath direction to send/receive.
```

Explanation:

- **unit(cfg)#09:13:55 CFXS > [EP PHONE] Change state to RINGING.**: The context CS interface changes the state to RINGING, that means it has accepted the call.
- **09:13:55 FXS > [0 1] Set state to 'Ringing'**: The port begins to ring on the analog line. This means that the binding of the port to the context CS interface is correct, and your phone should now be ringing.
- If the phone does not start to ring, revise the binding of the port to the context CS interface, and the connectors of the POTS line.

- **09:14:02 CFXS > [EP PHONE] Change state to CONNECTED.**: The phone went off-hook and thus accepted the call. The state of the CS interface goes to CONNECTED. The call is now established.

Debugging H.323 Signaling

Overview: H.323 debug monitors

	Command	Purpose
	***unit*#debug gateway h323 signaling**	Prints all signaling operations on H.323 interfaces.
	***unit*#debug gateway h323 error**	General purpose error monitor of H.323. Always enable this when debugging H.323.
	***unit*#debug gateway h323 datapath**	Prints operations on the H.323 part of the voice data path. Use this monitor if you experience problems in the data path (no speech connectivity, speech only in one direction). This monitor is not needed to debug signaling.
	***unit*#show gateway h323 status**	Shows the state of the H.323 gateway. Among others: enabled/disabled, RAS registration state.
	unit*#show gateway h323 call** *name* **[detail** *detail]**	Shows all calls ongoing on the H.323 interface on context CS with name *name*.
	unit*#show call-control provider** *name* **[detail** *detail]**	Shows the status of an H.323 interface in context CS (call control part of H.323 signaling). Shown are all calls currently ongoing on the interface, with their call states, signaling peers and voice data path parameters. *name* is the name of the H.323 interface. Use *detail* 5 for most verbose output.

See also section "Troubleshooting" on page 558 for further informations on debugging H.323. There are more than 40 debug monitors, such as *q931*, *ras*, or *signaling*. For a list of all available debug monitors, refer to the CLI online help.

Verify an incoming call

Make sure an incoming call is entering correctly context CS. In this example, a VoIP gateway or VoIP phone on the network makes a call to your unit using a gatekeeper. The debug output below shows a normal (working) call setup sequence

```
unit#enable
unit#configure
unit(cfg)#debug gateway h323 error
unit(cfg)#debug gateway h323 signaling
unit#debug gateway h323 ras
unit(cfg)#show gateway h323 status
H.323 Gateway: h323
===================

 State:                                UP
 Stack Handle:                         0x193ce44

 RAS Engine
```

```
 ----------

  State:                                REGISTERED
  Gatekeeper:                           172.16.32.51/1719

 Allocated Endpoints:                   0
 Allocated RAS Engines:                 1
 Allocated Control Channels:            0
 Allocated Outgoing Logical Slowstart Channels: 0
 Allocated Outgoing Logical Faststart Channels: 0
 Allocated Incoming Logical Channels:   0
unit#
unit#00:29:03 H323 > [EP h323-00c13dc0] Stack: Allocated new call: 0x00be56b0
00:29:03 H323 > Provider node_33: Added endpoint h323-00c13dc0
00:29:03 H323 > [EP h323-00c13dc0] Stack: Dial to remote terminal
00:29:03 H323 > [EP h323-00c13dc0]   Destination Address:             TEL:60,60
00:29:03 H323 > [EP h323-00c13dc0]   Source Address:                  TEL:50,50
00:29:03 H323 > [EP h323-00c13dc0]   Presentation Indicator:          Presentation
allowed
00:29:03 H323 > [EP h323-00c13dc0]   Screening Indicator:            User provided,
not screened
00:29:03 H323 > [EP h323-00c13dc0]   Information Transfer Capability: Speech
00:29:03 H323 > [EP h323-00c13dc0]   Display:                         50
00:29:03 H323 > [EP h323-00c13dc0]   User-User:
00:29:03 H323 > [EP h323-00c13dc0] Set state to TERMINAL TRYING
00:29:03 H323 > [EP h323-00c13dc0] State: TERMINAL TRYING, Call Event: PEER CON-
NECTED
00:29:03 H323 > [EP h323-00c13dc0] State: TERMINAL TRYING, Call Event: PROGRESS
00:29:03 HRAS > Stack: Received Admission Confirm
00:29:03 H323 > [EP h323-00c13dc0] Stack: State: DIALTONE
00:29:03 H323 > [EP h323-00c13dc0] Call-ID is
0213:4d80:eacb:11e0:2eee:0030:2b00:1e0e
00:29:03 H323 > [EP h323-00c13dc0] Stack: Received Incomplete-Address Indication
00:29:03 H323 > [EP h323-00c13dc0] Stack: State: RINGBACK
00:29:03 H323 > [EP h323-00c13dc0] Stack: Received ALERTING
00:29:03 H323 > [EP h323-00c13dc0]   Progress Indicator: (none)
00:29:03 H323 > [EP h323-00c13dc0] State: TERMINAL TRYING, Call Event: TERMINAL
ALERTING
00:29:03 H323 > [EP h323-00c13dc0] Set state to TERMINAL ALERTING
00:29:08 H323 > [EP h323-00c13dc0] Stack: State: CONNECTED (CALL-SETUP)
00:29:08 H323 > [EP h323-00c13dc0] State: TERMINAL ALERTING, Call Event: TERMINAL
CONNECTED
00:29:08 H323 > [EP h323-00c13dc0] Set state to CONNECTED
00:29:08 H323 > [EP h323-00c13dc0] Stack: Send STATUS INQUIRY
00:29:08 H323 > [EP h323-00c13dc0] Stack: Opening the H.245 control-channel
00:29:08 H323 > [EP h323-00c13dc0] State: CONNECTED, Call Event: PROGRESS
00:29:08 H323 > [EP h323-00c13dc0] Stack: Received STATUS. Audit successful
00:29:08 H323 > [EP h323-00c13dc0] Channel State: IDLE, Channel Event: CONTROL-UP
00:29:08 H323 > [EP h323-00c13dc0] Set channel state to OPENING
00:29:08 H323 > [EP h323-00c13dc0] State: CONNECTED, Call Event: PROGRESS
00:29:08 H323 > [EP h323-00c13dc0] Stack: State: CONNECTED (CALL)
00:29:08 H323 > [EP h323-00c13dc0] Stack: New Incoming Logical Channel: 00bcca48
00:29:08 H323 > [EP h323-00c13dc0] Channel State: OPENING, Channel Event: MODE-UP
00:29:08 H323 > [EP h323-00c13dc0] Set channel state to UP
```

Explanation:

- First the state of the gateway is checked. The state is “UP”, and the RAS engine is “REGISTERED”, which is OK.
- The line **`00:29:03 H323 > [EP h323-00c13dc0] Stack: Allocated new call: 0x00be56b`** tells that there is a new call incoming from H.323. This means that transport layer is OK. If there is no debug output at all, try to use **debug gateway h323 tpktchan**, which monitors all H.323 socket TCP traffic.
- The line **`00:29:03 H323 > [EP h323-00c13dc0] Destination Address: TEL:60,60`** and below show the calling and called party properties. In the present case, the called party number is 60.
- **`00:29:03 HRAS > Stack: Received Admission Confirm`** says that the gatekeeper allowed us to accept the call. Continue debugging call routing if the call does not reach its destination.

Verify an outgoing call

Make sure an outgoing call leaves correctly on text CS and the device. In this example, a terminal on the gateway to debug wants to make a call towards a H.323 VoIP network using a gatekeeper. The debug output below shows a normal (working) call setup sequence

```
unit#enable
unit#configure
unit(cfg)#debug gateway h323 error
unit(cfg)#debug gateway h323 signaling
unit#debug gateway h323 ras
unit(cfg)#show gateway h323 status
H.323 Gateway: h323
===================

 State:                                UP
 Stack Handle:                         0x193ce44

 RAS Engine
 ----------

  State:                               REGISTERED
  Gatekeeper:                          172.16.32.51/1719

 Allocated Endpoints:                  0
 Allocated RAS Engines:                1
 Allocated Control Channels:           0
 Allocated Outgoing Logical Slowstart Channels: 0
 Allocated Outgoing Logical Faststart Channels: 0
 Allocated Incoming Logical Channels:  0
unit#
unit#01:00:10 H323 > [EP h323-00c07230] Stack: Allocated new call: 0x00be5968
01:00:10 H323 > Provider node_33: Added endpoint h323-00c07230
01:00:10 H323 > [EP h323-00c07230] Stack: Dial to remote terminal
01:00:10 H323 > [EP h323-00c07230]   Destination Address:            TEL:60,60
01:00:10 H323 > [EP h323-00c07230]   Source Address:                 TEL:50,50
01:00:10 H323 > [EP h323-00c07230]   Presentation Indicator:         Presentation
allowed
01:00:10 H323 > [EP h323-00c07230]   Screening Indicator:           User provided,
not screened
01:00:10 H323 > [EP h323-00c07230]   Information Transfer Capability: Speech
```

```
01:00:10 H323 > [EP h323-00c07230]   Display:                          50
01:00:10 H323 > [EP h323-00c07230]   User-User:
01:00:10 H323 > [EP h323-00c07230] Set state to TERMINAL TRYING
01:00:10 H323 > [EP h323-00c07230] State: TERMINAL TRYING, Call Event: PEER CON-
NECTED
01:00:10 H323 > [EP h323-00c07230] State: TERMINAL TRYING, Call Event: PROGRESS
01:00:10 HRAS > Stack: Received Admission Confirm
01:00:10 H323 > [EP h323-00c07230] Stack: State: DIALTONE
01:00:10 H323 > [EP h323-00c07230] Call-ID is
0213:4d8d:13ba:1db8:2eef:0030:2b00:1e0e
01:00:11 H323 > [EP h323-00c07230] Stack: Received Incomplete-Address Indication
01:00:11 H323 > [EP h323-00c07230] Stack: State: RINGBACK
01:00:11 H323 > [EP h323-00c07230] Stack: Received ALERTING
01:00:11 H323 > [EP h323-00c07230]   Progress Indicator: (none)
01:00:11 H323 > [EP h323-00c07230] State: TERMINAL TRYING, Call Event: TERMINAL
ALERTING
01:00:11 H323 > [EP h323-00c07230] Set state to TERMINAL ALERTING
01:00:14 H323 > [EP h323-00c07230] Stack: State: CONNECTED (CALL-SETUP)
01:00:14 H323 > [EP h323-00c07230] State: TERMINAL ALERTING, Call Event: TERMINAL
CONNECTED
01:00:14 H323 > [EP h323-00c07230] Set state to CONNECTED
01:00:14 H323 > [EP h323-00c07230] Stack: Send STATUS INQUIRY
01:00:14 H323 > [EP h323-00c07230] Stack: Opening the H.245 control-channel
01:00:14 H323 > [EP h323-00c07230] State: CONNECTED, Call Event: PROGRESS
01:00:14 H323 > [EP h323-00c07230] Stack: Received STATUS. Audit successful
01:00:14 H323 > [EP h323-00c07230] Channel State: IDLE, Channel Event: CONTROL-UP
01:00:14 H323 > [EP h323-00c07230] Set channel state to OPENING
01:00:14 H323 > [EP h323-00c07230] State: CONNECTED, Call Event: PROGRESS
01:00:14 H323 > [EP h323-00c07230] Stack: State: CONNECTED (CALL)
01:00:14 H323 > [EP h323-00c07230] Stack: New Incoming Logical Channel: 00bccd68
01:00:14 H323 > [EP h323-00c07230] Channel State: OPENING, Channel Event: MODE-UP
01:00:14 H323 > [EP h323-00c07230] Set channel state to UP
```

Explanation:

- First the state of the gateway is checked. The state is *UP*, and the RAS engine is *REGISTERED*, which is OK.
- The line **`01:00:10 H323 > [EP h323-00c07230] Stack: Allocated new call: 0x00be5968`** tells that there is a new call incoming from H.323. This means that transport layer is OK. If there is no debug output at all, try to use **debug gateway h323 tpktchan**, which monitors all H.323 socket TCP traffic.
- The line **`01:00:10  H323 > [EP h323-00c07230] Destination Address: TEL:60,60`** and below show the calling and called party properties. In the present case, the called party number is *60*.
- **`00:29:03 HRAS > Stack: Received Admission Confirm`** says that the gatekeeper allowed us to dial to H.323 network with the given call parameters. If you can't see any H.323 SETUP message on the net, or if the called gateway does not indicate any H.323 message received, use **debug gateway h323 tpktchan** to verify that a packet is being sent.

Debugging SIP signaling

Mode: Administrator Execution

Step	Command	Purpose
1	[*node*]**#debug context sip-gateway data-path [detail** *<level>***]**	Logs information related to the media channels.
	[*node*]**#debug context sip-gateway error [detail** *<level>***]**	Logs all errors detected within the Context SIP Gateway.
	[*node*]**#debug context sip-gateway registration [detail** *<level>***]**	Logs information about user registration activities.
	[*node*]**#debug context sip-gateway signaling [detail** *<level>***]**	Logs call signaling related information.
	[*node*]**#debug context sip-gateway transport [detail** *<level>***]**	Logs all SIP messages sent or received over the IP network.

Verify an incoming call

Make sure that an incoming call from the SIP network enters correctly context CS. The following sequence shows a working call setup.

```
unit(cfg)#debug gateway sip error
unit(cfg)#debug gateway sip transport
unit(cfg)#18:53:40  SIP_TR> Received INVITE sip:50@172.16.32.32 SIP/2.0
18:53:40 SIP_TR> Sent SIP/2.0 100 Trying
18:53:40 SIP_TR> Sent SIP/2.0 180 Ringing
18:53:43 SIP_TR> Sent SIP/2.0 200 OK
18:53:43 SIP_TR> Received ACK sip:50@172.16.32.32:5060 SIP/2.0
```

Explanation:

- The line **`18:53:40 SIP_TR> Received INVITE sip:50@172.16.32.32 SIP/2.0`** indicates that the *INVITE* message has been received. This means that the SIP network is functional
- **`18:53:40 SIP_TR > Sent SIP/2.0 100 Trying`** and **`18:53:40 SIP_TR> Sent SIP/2.0 180 Ringing`** indicate that responses are sent back to the SIP network. This means that the call routing is working correctly, and the call has found its destination on the gateway that is debugged. If there are no responses, or a negative response, continue debugging call routing and the destination protocol.

Verify an outgoing call

Make sure that an outgoing call from context CS leaves correctly to the SIP network. The following sequence shows a working call setup.

```
unit(cfg)#debug gateway sip error
unit(cfg)#debug gateway sip transport
unit(cfg)#18:59:07 SIP_TR> Sent INVITE sip:60@172.16.32.33 SIP/2.0
18:59:07 SIP_TR> Received SIP/2.0 100 Trying
18:59:07 SIP_TR> Received SIP/2.0 180 Ringing
18:59:10 SIP_TR> Received SIP/2.0 200 OK
18:59:10 SIP_TR> Sent ACK sip:60@172.16.32.33:5060 SIP/2.0
```

Explanation:

- The line **`18:59:07 SIP_TR> Sent INVITE sip:60@172.16.32.33 SIP/2.0`** indicates that the *INVITE* was sent. Thus, call routing worked in context CS and the message left to the SIP network.
- **`18:59:07 SIP_TR > Received SIP/2.0 100 Trying`** indicate that responses are received from the SIP network. This means that IP connectivity is OK and the remote gateway can be reached. If there are no responses, or negative ones, continue debugging the remote SIP gateway.

Using SmartWare's internal call generator

The SmartWare has a powerful internal call generator that creates calls from the center of context CS. It is very useful to debug call signaling or call routing problems to verify the correct working of one call signaling protocol at a time. Calls can be placed towards any interface on context CS, or any routing table within context CS.

	Command	Purpose
	call *calling-party* **{accept | alerting | dial|drop | inband-info | offer-state | proceeding | reject-all | resume | suspend | user-input } ....**	Manipulates calls locally generated from the center of context CS. To create a call, use "call *calling-party* dial *called-party*", to manipulate this call afterwards use the same calling party. See the examples below for usage.

Example: Debug ISDN protocol using the call generator. Create a call from context CS to an ISDN interface called TERMINAL.

```
unit(cfg)#debug ccisdn signaling
unit(cfg)#debug ccisdn error
unit(cfg)#debug isdn error
unit(cfg)#call 55 dial 50 dest-interface TERMINAL
unit(cfg)#19:17:38  ICC   > [TERMINAL] Added endpoint TERMINAL-00df2760
19:17:38  ICC   > [TERMINAL] NEW CALL. Allocated Endpoint TERMINAL-00df2760
19:17:38  ICC   > [TERMINAL-00df2760] >> SETUP (DSS1 Ntwk)
  Bearer capability : speech - CCITT
    circuit mode - 64kBit/s - G.711 A-law
  Calling party number : 55
    unknown number - unknown numbering plan
    presentation allowed - user provided not screened
  Called party number : 50
    unknown number - unknown numbering plan
  High layer compatibility : telephony
    CCITT

19:17:38  ICC   > [TERMINAL-00df2760] Set state to CALL PRESENT
19:17:38  ICC   > [TERMINAL-00df2760] State: CALL PRESENT, Event: PEER CONNECTED
19:17:38  ICC   > [TERMINAL] << Message: primitive=31
19:17:38  ICC   > [TERMINAL-00df2760] << ALERTING (DSS1 Ntwk)

19:17:38  ICC   > [TERMINAL-00df2760] State: CALL PRESENT, Event: TERMINAL ALERTING
IND
19:17:38  ICC   > [TERMINAL-00df2760] Set state to CALL RECEIVED
19:17:44  ICC   > [TERMINAL] << Message: primitive=33
19:17:44  ICC   > [TERMINAL-00df2760] << CONNECT (DSS1 Ntwk)
  Connected number : 50
    unknown number - unknown numbering plan
```

```
    presentation allowed - user provided not screened

19:17:44  ICC   > [TERMINAL-00df2760] State: CALL RECEIVED, Event: TERMINAL CONNECT
IND
19:17:44  ICC   > [TERMINAL-00df2760] Set state to ACTIVE
19:17:44  ICC   > [TERMINAL-00df2760] >> CONNECT ACKNOWLEDGEMENT (DSS1 Ntwk)
unit(cfg)#
unit(cfg)#call 55 drop
unit(cfg)#19:19:29  ICC   > [TERMINAL-00df2760] State: ACTIVE, Event: PEER RELEASED
19:19:29  ICC   > [TERMINAL-00df2760] Set state to DISCONNECT INDICATION
19:19:29  ICC   > [TERMINAL-00df2760] >> DISCONNECT (DSS1 Ntwk)
  Cause : normal call clearing
    private network serving local user - CCITT - Q.931
  Progress indicator : inband information available
    private network serving local user - CCIT
```

Explanation:

- **`unit(cfg)#call 55 dial 50 dest-interface TERMINAL`**: Dial the number *50*, which calling-party 55 to the interface *TERMINAL*. If a phone connected to the port that binds to interface *TERMINAL* has MSN 50 configured, it will start to ring. This is the case in our example.
- You just have verified that ISDN layer1–3 and the context CS interface *TERMINAL* are configured so that a call with called-party 50 can be handled. If this is not the case (no response or a *RELEASE* message), continue debugging ISDN signaling.
- **`unit(cfg)#call 55 drop`**: Drops the call initiated with the **dial** command.

You can proceed as in this example with any other context CS interface, also for VoIP protocols like SIP and H.323.

Debugging voice data

There are several debug monitors that can help identify problems in VoIP connections. The most common VoIP problems are: voice quality problems (dropouts), fax transmission errors, no establishment of voice connection, and wrong tone or playback. Dependent on the SmartNode devices, the output of the different Media Gateway monitors can differ.

An overview of all available Media Gateway debug monitors is given below. Some more specific examples for debugging cases follow after that.

Overview: Media Gateway debug monitors

[no] debug media-gateway dejitter	Displays changes to the settings of the dejitter buffer, exceptions (underrun, overrun, packet drops) and size changes. Usage: To investigate problems related to voice quality, voice packet payload sizes, delays, jitter
[no] debug media-gateway control [detail level]	Displays control activities on the Data Path (path of voice/fax data packets within your unit): State changes, tone start/stop, DTMF playback/detection, fax/modem detection. Usage: To investigate problems with voice connections, DTMF, tone playback, fax and modem transmissions.

[no] debug media-gateway rtp	Displays RTP related call parameters at call setup: local/remote IP address and port, SSRC. During operation, displays periodically updated statistics containing the number of sent and received packets, the number of lost packets. Usage: To verify that RTP packets are sent/received, and to debug network quality issues (lost packets).
[no] debug media-gateway switch	Displays control activities on the TDM part of the Data Path. Usage: To investigate problems with hair pinning or timeslot switching.
[no] debug media-gateway dsp	Displays control activities on the DSP including all call parameters (e.g. the used codec). Usage: To document the DSP parameters used for each call setup.
[no] debug media-gateway fax-data	Traces detected Modem and Fax tones. Traces the flow of T.30 communication states between the fax machines. Decodes the IFP (Internet Fax Protocol) elements within T.38 packets. Displays T.38 related call parameters and operational errors like lost packets. Tracks the operation of the dejitter buffer used for T.38 Fax transmissions. Usage: To investigate problems with T.38 Fax and Modem transmissions in general.
[no] debug media-gateway error	Displays detected errors over all the different parts of the Media Gateway.
[no] debug media-gateway all	Enables/Disables all Media Gateway monitors

Depending on the type of problem, some debug monitors are more useful than others. Try to avoid enabling all monitors at the same time, as this generates a lot of output and can degrade system performance. The following examples show some typical debug cases and what monitors should be switched on in these cases.

Example: Debugging voice connections

Symptoms: Voice quality is bad (dropouts), the voice connection is only established in one direction or not at all, there is only noise instead of voice, no tones are played (e.g. dialtone).

Prerequisite: The call is established from a signaling point of view (see chapters Debug H.323 Data and Debug Session Control Data).

Use the following debug monitors:

Step	Command	Purpose
1	***unit*#debug media-gateway rtp**	**Enable the RTP/RTCP debug monitor**
2	***unit*#debug media-gateway dejitter**	**Enable the dejitter buffer monitor.**

Example: Debugging Fax connections

Symptoms: Fax transmission starts but is interrupted and the Fax machines terminate with an error message, Fax transmission does not start at all, the unit's firmware does not detect Fax.

Prerequisite: Fax transmission is configured for T.38 relay in the voip profile. The call is established from a signaling point of view (see section "Debugging voice data" on page 640).

Attention: Special signaling procedures are used for the transition between voice and fax data transmission. It may be that the initial call setup is correct, but that the signaling to T.38 over H.323 is faulty. The session-control and H.323 signaling monitors (see Debug Session Control Data, and Debug H.323 Data) might also be helpful to identify these cases.

Use the following debug monitors if you assume that signaling is OK:

Step	Command	Purpose
1	***unit*#debug media-gateway dsp**	Enable the dsp event monitor.
2	***unit*#debug media-gateway fax-data**	Enable the Fax/Modem event monitor
3	***unit*#debug media-gateway control**	Enable the control monitor

Check system logs

See section "Displaying the system logs" on page 97.

How to submit trouble reports to Patton

Due the wealth of functionality and complexity of the products there remains a certain number of problems, either pertaining to the Patton product or the interoperability with other vendor's products.

If you have a problem for which you need supplier help please prepare and send the following information:

- **Problem description**—Add a description of the problem, if possible together with applicable augmented information with a diagram of the network setup (with Microsoft tools).
- **Running configuration and software and hardware version information**—With the Command Line Interface commands **show running-config** and **show version** you can display the currently active configuration of the system (in a Telnet and/or console session). Adding to the submitted trouble report will help us analyze the configuration and preclude possible configuration problems.

  In the unlikely case of a suspected hardware problem also submit the serial number of your unit(s) and/or interface cards.
- **Event logs**—Add the system event logs, which you can display with the Command Line Interface commands **show log** and **show log supervisor**. To ensure that the logs are useful, it is necessary to set upon start up the clock to actual date and time (by hand or by enabling SNTP client)
- **Your location**—For further enquiries please add your E-mail address and phone number.

If possible, add the following information in addition to the above:

- **Logs of protocol monitors**—Protocol traces contain a wealth of additional information which may be very helpful in finding or at least pinpointing the problem. Various protocol monitors with different levels of detail are an integral part of the firmware and can be started (in a Telnet and/or console session) individually (**debug** command).

  Note In order to correlate the output of different protocol monitors (e.g. ISDN signaling and gateway SIP signaling), run the monitors concurrently. You

can do this either in the same Telnet session, or using different Telnet sessions.

- **Network traffic traces**—In certain cases it may be helpful to have a trace of the traffic on the IP network in order to inspect packet contents. Please use one of the following tools (supporting trace file formats which our tools can read):
  - Network Associates Sniffer—Details are available at www.sniffer.com
  - TTC Firebird—Details are available at www.ttc.com
  - Ethereal—Details are available at www.ethereal.com (freeware)

When possible, submit the package of trouble report files by mail to the following address: support@patton.com (use fax only in exceptional cases).

Appendix A **Terms and definitions**

Chapter contents

Introduction 645
SmartWare architecture terms and definitions 645

Introduction

This chapter contains the terms and their definitions that are used throughout this *SmartWare Software Configuration Guide*. This guide contains many terms that are related to specific networking technologies areas such as LAN protocols, WAN technologies, routing, Ethernet, and Frame Relay. Moreover various terms are related to telecommunication areas.

SmartWare architecture terms and definitions

Term or Definition	Meaning
Administrator	The person who has privileged access to the CLI.
Application Download	A application image is downloaded from a remote TFTP server to the persistent memory (flash:) of a SmartNode.
Application Image	The binary operation code stored in the persistent memory (flash:) of a SmartNode.
Batchfile	Script file containing instructions to download one or more software component from a TFTP server to the persistent memory (flash: or nvram:) of a SmartNode.
Bootloader	The bootloader is a "mini" application performing basic system checks and starting the application image. The bootloader also provides minimal network services allowing the SmartNode to be accessed and upgraded over the network even if the application image should not start. The bootloader is installed in the factory and is in general never upgraded.
Bootloader Image	The binary code of the Bootloader stored in the persistent memory (flash:) of a SmartNode.
Bootstrap	The starting-up of a SmartNode, which involves checking the Reset button, loading and starting the application image, and starting other software modules, or—if no valid application image is available—the bootloader.
Build	The released software is organized as builds. Each build has its unique identification. A build is part of a release and has software bug fixes. See also *release*.
Call Routing	Calls through SmartNode can be routed based on a set of routing criteria. See also Session Router.
Call Signaling	The call signaling specifies how to set up a call to the destination SmartNode or 3rd party equipment.
Circuit	A communication path between two or more devices.
Circuit Port	Physical port connected to a switching system or used for circuit switching.
Circuit Switching	The switching system in which a dedicated physical circuit path must exist between the sender and the receiver for the duration of the call. Used in the conventional telephone network.
Codec	Abbreviation for the word construct Coder and Decoder. Voice channels occupy 64 kbps using PCM (pulse code modulation) coding. Over the years, compression techniques were developed allowing a reduction in the required bandwidth while preserving voice quality. Such compression techniques are implemented within a Codec.

Term or Definition	Meaning
Comfort Noise	Comfort noise is generated at the remote end of the silent direction to avoid the impression that the connection is dead. See also *Silence Compression*.
Command Line Interface	An interface that allows the user to interact with the SmartWare operating system by entering commands and optional arguments. Other operating systems like UNIX or DOS also provide CLIs.
Configuration Download	A configuration file is downloaded from a remote TFTP server via TFTP to the persistent memory (nvram:) or volatile memory (system:)of a SmartNode.
Configuration File	The configuration file contains the CLI commands, which are used to configure the SmartNode.
Configuration Server	A central server used as a store for configuration files, which are downloaded to or uploaded from a SmartNode using TFTP.
Configuration Upload	A configuration file is uploaded from the persistent memory (nvram:) or volatile memory (system:) of a SmartNode via TFTP to a TFTP server.
Context	Represents one specific networking technology or protocol, e.g. IP or circuit switching.
Data Port	Physical port connected to a network element or used for data transfer.
Dejitter Buffer	The element used to compensate variable network delays. Storing packets in a dejitter buffer before they are transferred to the local ISDN equipment, e.g. telephone, a variable delay is converted into a fixed delay, giving voice a better quality. See also *Jitter*.
Digit Collection	Some devices (PBX, ISDN network, remote gateways and gatekeepers) may require bloc sending of the dialed number. Digit collection collects the overlap dialed digits and forwards them in a single call setup message
Driver Software Download	A driver software image is downloaded from a remote TFTP server to the persistent memory (flash:) of a SmartNode.
Driver Software Image	The software used for peripheral chips on the main board and optional PMC interface cards is stored in the persistent memory (flash:) of a SmartNode.
DTMF Relay	DTMF relay solves the problem of DTMF distortion by transporting DTMF tones over low-bit-rate codecs out-of-band or separate from the encoded voice stream
Echo Canceller	Some voice devices unfortunately have got an echo on their wire. Echo cancellation provides near-end echo compensation for this device.
Factory Configuration	The factory configuration (factory-config) represents the system default settings and is stored in the persistent memory (nvram:) of a SmartNode.
Fast Connect	A "normal" call setup with H.323 requires several TCP segments to be transmitted, because various parameters are negotiated. Since a normal call setup is often too slow, fast connect is a new method of call setup that bypasses some usual steps in order to make it faster.
Flash Memory	Persistent memory section of a SmartNode containing the Application Image, Bootloader Image and the driver software Image.

Term or Definition	Meaning
flash:	A region in the persistent memory of a SmartNode. See also *flash memory*.
Gatekeeper	Gatekeepers manage H.323 zones, which are logical collections of devices such as all H.323 devices within an IP subnet. For example, gatekeepers provide address translation (routing) for the devices in their zone.
Gateway	A gateway refers to a special purpose component that connects two contexts of different types, For example, the CS and the IP context. It handles connections between different technologies or protocols. SmartWare includes an H.323 and SIP gateway.
H.323	ITU-T recommendation H.323 describes terminals, equipment and services for multimedia communication over Local Area Networks (LAN) which do not provide a guaranteed quality of service. H.323 terminals and equipment may carry real-time voice, data and video, or any combination, including video telephony.
H.323 RAS	H.323 registration authentication service (RAS) is a sub protocol of H.323. The RAS signaling protocol performs registration, admissions, and bandwidth changes and disengage procedures between the VoIP gateway and the gatekeeper.
High-Pass Filter	A high-pass filter is normally used to cancel noises at the voice coder input. See also *post filter*
Host	Computer system on a network. Similar to node, except that host usually implies a PC or workstation, whereas node generally applies to any networked system, including access servers and routers. See also *node*.
Hostname	Name given to a computer system, e.g. a PC or workstation.
Hunt Group	In SmartNode terminology, a hunt groups allows you to apply the interface configuration to multiple physical ports. Within the hunt groups free channels for outgoing calls are hunted on all available ports. In general a hunt group represents a group of trunk lines as used for direct dialing in (DDI).
Interface	An interface is a logical construct that provides higher-layer protocol and service information. An Interface is configured as a part of a context, and is independent of a physical port or circuit.
Interface Card	An optional plug-in card offering one or more ports of a specific physical standard for connecting the SmartNode to the outside world.
ISDN	Integrated Services Digital Network
ISDN Services	ISDN Services comprise voice, data, video and supplementary services. Supplementary services are services available in the ISDN network, such as calling line identification presentation (CLIP) or call waiting (CW). See also *Q.SIG*
Jitter	Jitter is the variation on packets arriving on a SmartNode. See also *dejitter buffer*.
Mode	The CLI is comprised of modes. There are two basic mode groups, the execution mode group and the configuration mode group.

Term or Definition	Meaning
Network Management System	System responsible for managing at least part of a network. An NMS is generally a reasonably powerful and well-equipped computer, such as an engineering workstation. NMSs communicate with agents to help keep track of network statistics and resources.
Node	Endpoint of a network connection or a junction common to two or more lines in a network. A Node can be a router, e.g. a SmartNode. Nodes, which vary in routing and other functional capabilities, can be interconnected. Node sometimes is used generically to refer to any entity that can access a network, and frequently is used interchangeably with device.
Nodename	Name given to a SmartNode or network element.
nvram:	Persistent memory section of a SmartNode containing the startup configuration, the factory configuration and used defined configurations.
Operator	The person who has limited access to the 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.
Port	A port represents a physical connector on the SmartNode.
Port Address	A port address can be assigned to a CS interface to realize a virtual voice tunnel between two nodes.
Post Filter	The voice decoder output is normally filtered using a perceptual post-filter to improve voice quality. See also *High-Pass Filter*.
POTS	Plain Old Telephone Service
Profile	A profile provides configuration shortcutting. A profile contains specific settings that can be used on multiple contexts, interfaces or gateways.
PSTN	Public Switched Telephone Network. Contains ISDN and POTS
Q.931 Tunneling	Q.931 tunneling is able to support ISDN services and Q.SIG over an IP network.
Q.SIG	ISDN Services comprise additional services for the Private ISDN network such as CNIP (Calling Name Identification Presentation), CNIR (Calling Name Identification Restriction) etc. See also *ISDN Services*.
Release	A software release describes the main voice and data feature set. It consists of a series of builds.
Routing Engine	The routing engine handles the basic IP routing.
Running Configuration	The currently running configuration (running-config), which is executed from the volatile memory (system:).
Session Router	Calls through SmartNode can be routed based on a set of routing criteria. The entity that manages call routing is called Session Router.
Session Initiation Protocol (SIP)	Used for setting up communications sessions (such as conferencing, instant messaging, and telephony) on the Internet
Silence Compression	Silence suppression (or compression) detects the silent periods in a phone conversation and stops the sending of media packets during this periods.

Term or Definition	Meaning
Startup Configuration	The startup configuration is stored in the persistent memory (nvram:) and is always copied for execution to the running configuration in the volatile memory (system:) after a system start-up.
Switching Engine	Part of the SmartNode hardware which allows software controlled circuit switching of circuit ports.
System Image	A collective term for application images and interface card driver software, excluding configuration files.
System Memory	The volatile memory, that includes the system: region, holding the running-config for the SmartWare during operation of a SmartNode.
system:	A region in the volatile memory of a SmartNode. See also *system memory*.
TFTP Server	A central server used for configuration up- and download, download of application and interface card driver software, that is accessed using TFTP.
tftp:	Identification of a remote storing location used for configuration up- and download, download of application and interface card driver software, that is accessed using TFTP.

Appendix B **Mode summary**

Chapter contents

Introduction 651

Introduction

Figure 1 on page 651, figure 2 on page 652, and figure 3 on page 653 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 654.

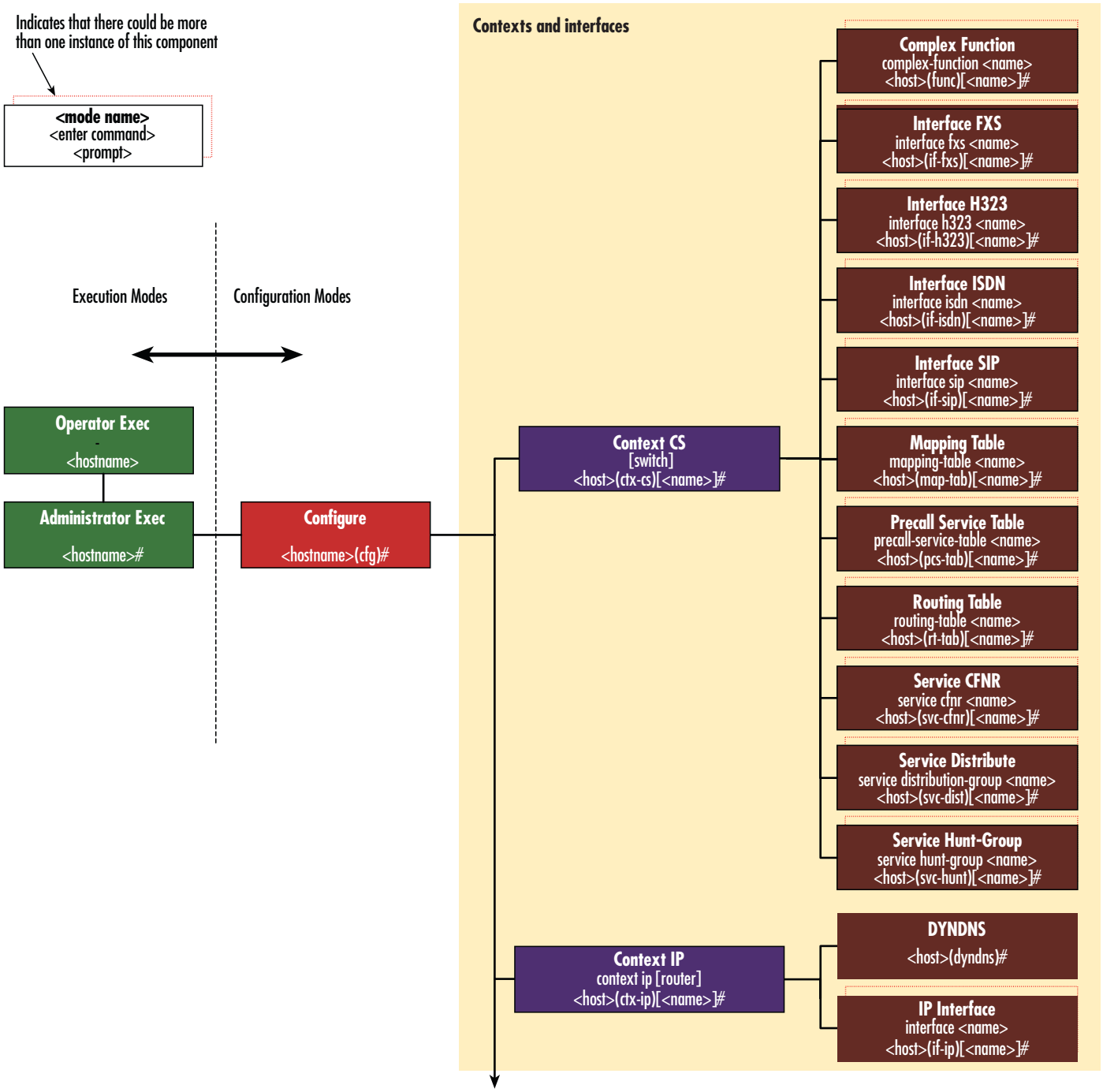


Figure 1. Mode overview, 1 of 3

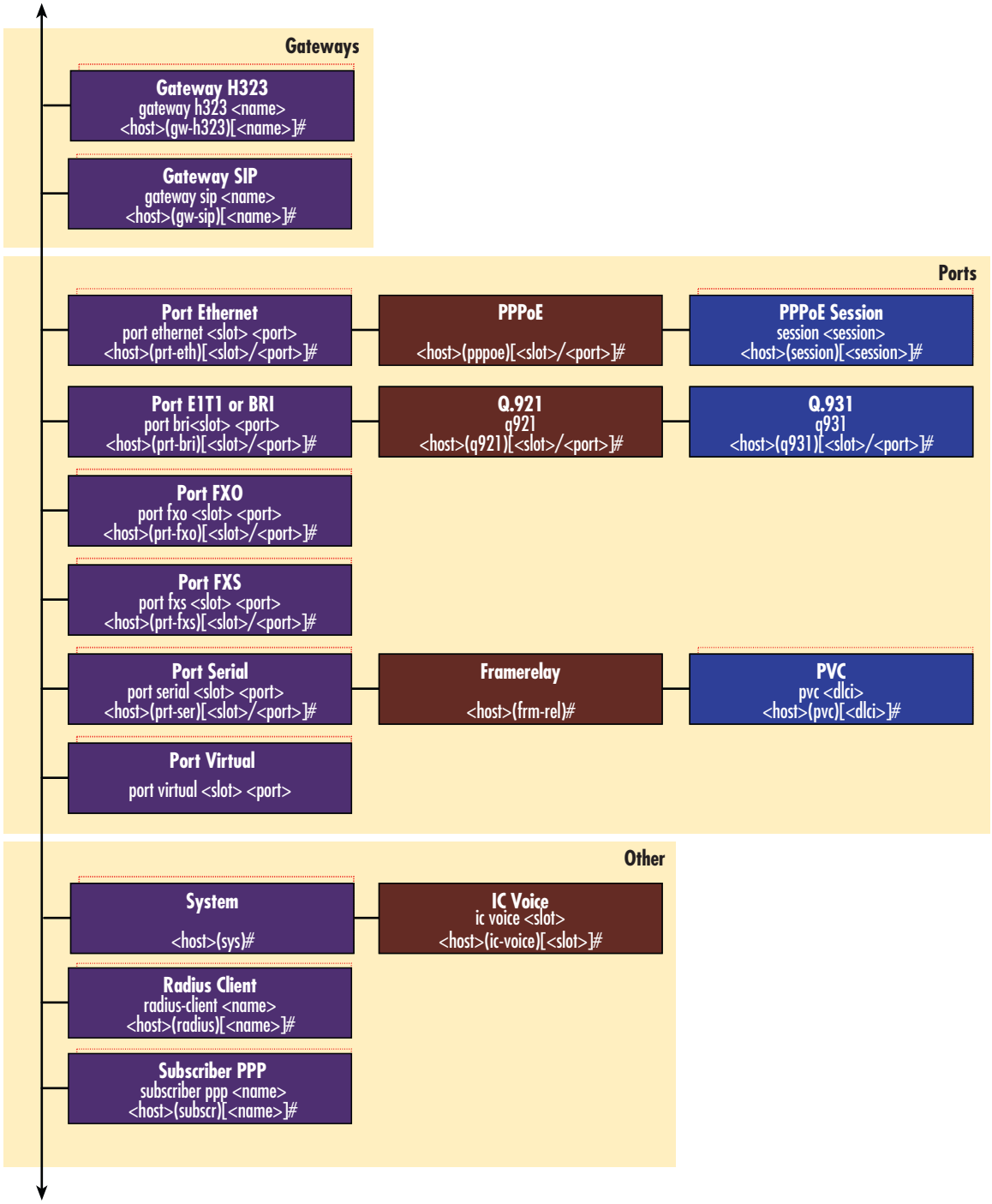


Figure 2. Mode Overview, 2 of 3

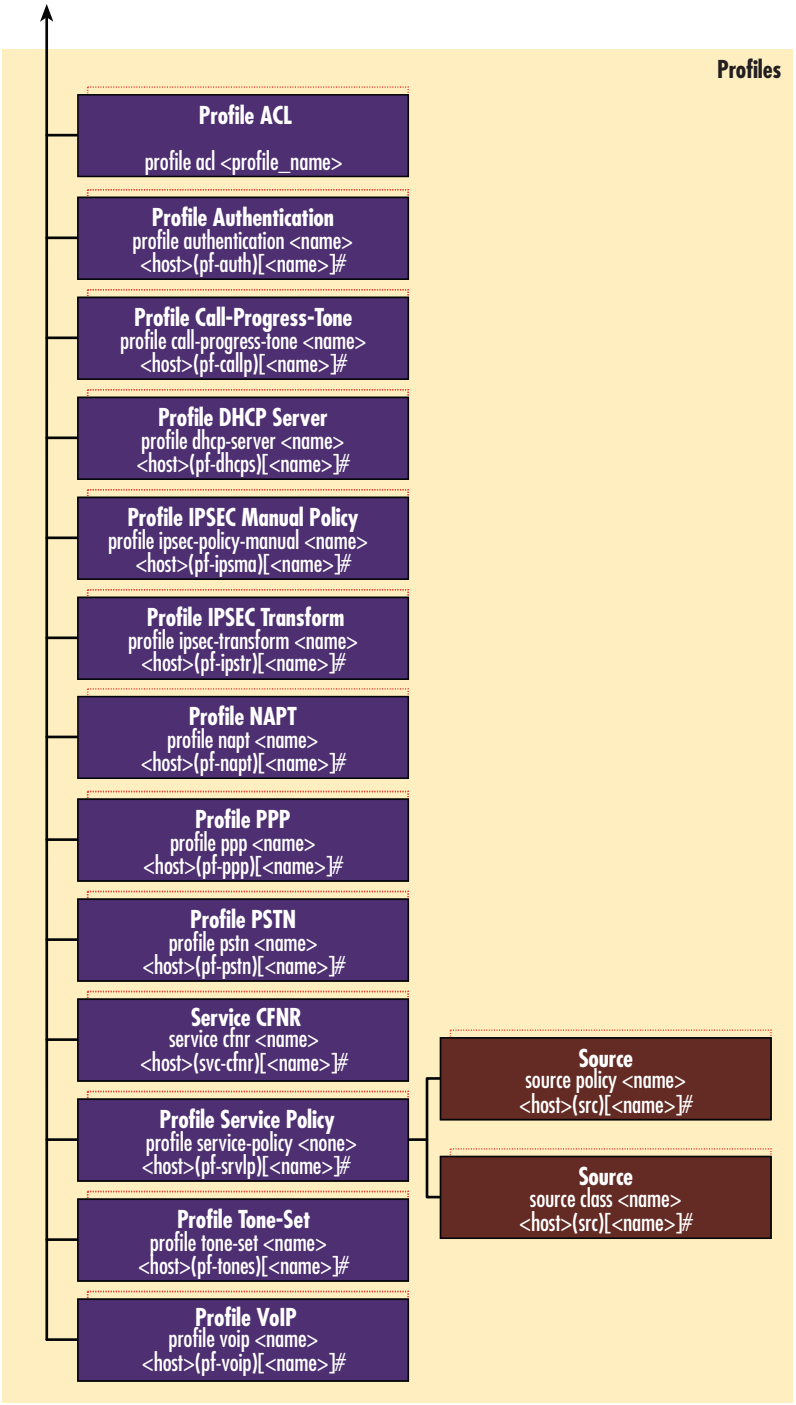


Figure 3. Mode Overview, 3 of 3

Appendix C **Command summary**

Chapter contents

Introduction 655
New Configuration Commands 656
Other 656
- Show help 656
- Show command history 656
- Restart system 656

Introduction

This chapter provides an overview of all CLI commands and modes available. It is organized as follows:

```
Mode Name
Enter Command
Command 1
…
Exit

Mode Name
…
```

Several commands contain a lot of parameters and arguments. The command syntax is described as follows:

- Arguments where you must supply the value are surrounded by <angle brackets>.
- Optional arguments within commands are shown in square brackets ([ ])
- Alternative parameters within commands are separated by vertical bars (|).
- Alternative but required parameters are shown within grouped braces ({ }) and are separated by vertical bars (|).

Command syntax is illustrated by an example in figure 4.

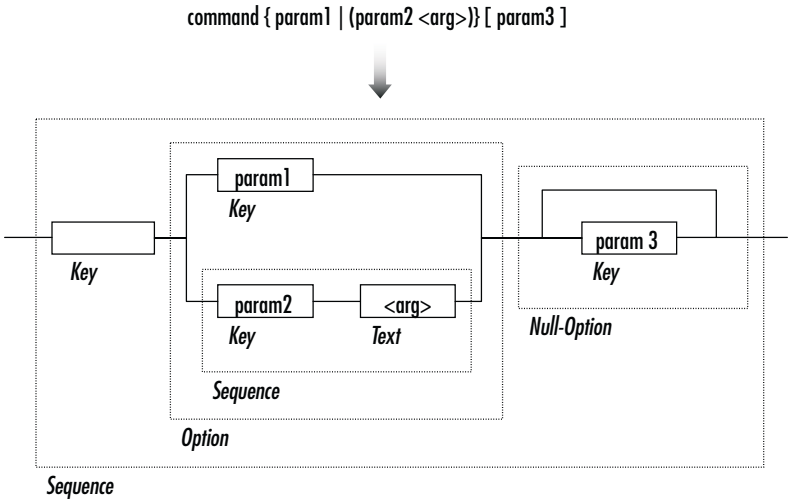


Figure 4. EBNF syntax

New Configuration Commands

The commands documented in the Release Note only cover new additions which are not yet included in the current revision of the Software Configuration Guide. You may download the release notes at www.patton.com/support.

Current Revision:

Document Number: 13211U8-001 Rev. D

Part Number: 07MSWR320_SCG

Revised: July 17, 2006

Other

Show help

Step	Command	Purpose
1	**help** [*topic*]	Shows command help.

Show command history

Step	Command	Purpose
1	**history**	Shows command history.

Use CTRL-N and CTRL-P to browse. The cursor keys (up, down) are not working.

Restart system

Step	Command	Purpose
1	**reset**	Restarts the system.

Appendix D **Internetworking terms & acronyms**

Chapter contents

Abbreviations.....658

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
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
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
M	
MIB II	Management Information Base II

Abbreviation	Meaning
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
RTP	Real-time Protocol
S	
S1	node-connection for Trunk Line

Abbreviation	Meaning
S2	node-connection for Subscriber Line
SAR	Segmentation and Reassembly
S-Bus	Subscriber Line (Connection) Bus
SCN	Switched Circuit Network
SCTP	Stream Control Transmission Protocol
SDSL	Symmetric Digital Subscriber Line
SGCP	Simple Gateway Control Protocol
SIP	Session Initiation Protocol.
SME	Small and Medium Enterprises
SNMP	Simple Network Management Protocol
SOHO	Small Office Home Office
SONET	Synchronous Optical Network
SS7	Signaling System No. 7
STM	SDH Transmission at 155 Mb/s
SVC	Switched Virtual Circuit
SW	Software
T	
TCP/IP	Transport Control Protocol/Internet Protocol
TE	Terminal Equipment
TFTP	Trivial File Transfer Protocol
U	
UBR	Unspecified Bit Rate
UD 64	Unrestricted Data 64 kb/s
UDP	User Datagram Protocol
V	
VBR	Variable Bit Rate
VCI	Virtual Channel Identifier
VoIP	Voice over Internet Protocol
VPI	Virtual Path Identifier
W	
WAN	Wide Area Network

Appendix E Used IP ports & available voice codecs

Chapter contents

Used IP ports 663
Available voice codecs 664

Used IP ports

Component	Port	Description
H.323	UDP 1719	RAS for gatekeeper connection
	TCP 1720	Call signaling port for H.323 (adjustable)
	UDP 4864…5118 (even numbers)	Voice data (RTP)
	UDP 4865…5119 (odd numbers)	Voice statistics (RTCP)
SIP	UDP 5060	Call signaling port for SIP (configurable)
	TCP 5060	Call signaling port for SIP (configurable)
	UDP 4864…5118 (even numbers)	Voice data (RTP)
	UDP 4865…5119 (odd numbers)	Voice statistics (RTCP)
NAPT	TCP 8000-15999	NAPT port range
Telnet	TCP 23	TCP server port
Webserver	TCP 80	TCP server port
DHCP	UDP 67	Source port DHCP Server
	UDP 68	Source port DHCP Client
TFTP	UDP 69	Control port of the TFTP Server (accessed by the TFTP Client in the SmartNode)

Available voice codecs

Protocol	Codec	Net Bandwidth per Call (kbps)	Min. Compression Delay (ms)	Used Bandwidth per Call (kbps, incl. IP header)	Usage
H.323 & SIP	G.711 A-law	64	10	96	Uncompressed, best voice quality, European audio-digitizing
	G.711 U-law	64	10	96	Uncompressed, best voice quality, American audio-digitizing
	G.723.1	6.3	30	17	Good voice quality at lowest bandwidth, like analog phone, acceptable delay
	G.729/ G.729a	8	10	40	Best relationship between voice quality and used bandwidth, low delay
	Transparent	64	10	96	Transparent ISDN data, no echo cancellation
	G.726	16, 24, 32, 40	20	32, 40, 48, 56	The G.726 is an ADPCM based codec, with small memory footprint but fairly high CPU time requirements. Also available in Cisco compatible mode.
	G.727	16, 24, 32	20	32, 40, 48	Embedded ADPCM. See also G.726.
	T.38	Variable			This is a fax codec, not a voice codec.