

# ***NexTone Multiprotocol Session Exchange (MSX)***

## ***Operations Guide***

*Release 4.3, Issue 1*

*May 11, 2007*

**NEX**TONE

Real time ■ Real communications ■ Real results

101 Orchard Ridge Drive, Suite 300  
Gaithersburg, MD 20878, USA

**© 2000-2007 and <sup>TM</sup> NexTone Communications, Inc.**

All rights reserved. This publication and its contents are proprietary to and a trade secret of NexTone Communications. No part of this publication may be reproduced, distributed or modified in any form or by any means without the written permission of NexTone Communications.

United States and foreign patents pending.

The information in this document is subject to change without notice. NexTone Communications makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.

#### **TRADEMARKS:**

NexTone Communications, NexTone, iServer, MSW, MSC, MSX, NARS, iView, iVMS, RSM, “Network without limits” and IntelliConnect all are trademarks of NexTone Communications. Other trademarks, marks, names, or product names referenced in this publication are the property of their respective owners.

All parts of this document and the information contained in it are protected by U.S and International intellectual property laws.

# Table of Contents

<b>Chapter 1: About This Guide .....</b>	<b>1</b>
Who Should Use This Guide .....	1
What's in this guide .....	1
Typographic conventions.....	2
What Documentation Is Required.....	3
How to Contact NexTone Support.....	4
<b>Chapter 2: What Does the MSX Do? .....</b>	<b>5</b>
Who should use this document .....	5
MSX components and services .....	6
<i>MSX configuration data</i> .....	6
<i>H.323 gatekeeper services</i> .....	6
<i>Inter-Working Function (IWF) services</i> .....	6
<i>Firewall Control Entity (FCE) services</i> .....	7
<i>NexTone Session Filter (NSF) services</i> .....	7
Real-time Session Manager (RSM) Components .....	8
<i>Real-time Session Manager (RSM)</i> .....	8
<i>RSM Lite™ and RSM Console™</i> .....	8
<b>PART 1:</b>	
<b>MSX SYSTEM - - - - -</b>	<b>9</b>
<b>Chapter 3: MSX Configuration.....</b>	<b>10</b>
Supported endpoint types.....	10
<i>Generic IP device</i> .....	11
<i>H.323 gateway, gatekeeper &amp; sgatekeeper</i> .....	11
<i>SIP gateway/SIP proxy</i> .....	13
<i>Softswitch/ENUM server/MSX</i> .....	14
<i>User account</i> .....	15
<i>Inoch server</i> .....	15
MSX global settings.....	16

Understanding MSX configuration interfaces .....	16
<i>Using RSM console</i> .....	16
Global configuration using nxconfig.pl .....	18
<i>Running the nxconfig.pl utility</i> .....	19
<i>servercfg attributes</i> .....	22
<i>Setting SIP parameters</i> .....	50
<i>Configuring optional H.323 parameters</i> .....	50
<i>Setting the global default ENUM domain</i> .....	50
<i>Enabling the Firewall Control Entity (FCE) feature</i> .....	50
<i>Setting QoS parameters</i> .....	51
<i>MSX peering and database replication parameters</i> .....	52
<i>MSX management interface configuration</i> .....	53
Global configuration using RSM Console .....	54
<i>Allow All Sources</i> .....	61
<i>Call hunting with multiple directory gatekeepers</i> .....	63
<i>Local Number Portability (LNP) support</i> .....	63
Procedure: add an endpoint .....	65
<i>Determining the device type</i> .....	65
<i>Master gatekeeper or peering gatekeeper</i> .....	66
<i>What to configure, and what to ask</i> .....	66
<i>Making an endpoint “sticky”</i> .....	78
Setting session limits on calls, by endpoint .....	79
<i>U-port group limits</i> .....	80
<i>Subnet CAC</i> .....	83
Limiting bandwidth by call .....	90
Setting call duration limits .....	90
<i>Maximum call duration</i> .....	90
<i>SIP Timer C</i> .....	91
Disabling media routing on specific endpoints .....	91
Enabling LNP for an endpoint .....	92
<i>CLI procedure</i> .....	92
<i>RSM Console procedure</i> .....	93
<b>Chapter 4: MSX Administration .....</b>	<b>96</b>
Introduction .....	96
Operating System maintenance .....	96
Backing up an MSX .....	96
<i>System file backup</i> .....	96
<i>Configuration Database Backup</i> .....	98

Starting and stopping the MSX.....	98
<i>Determining MSX status</i> .....	98
<i>Stopping the MSX</i> .....	99
<i>Starting the MSX</i> .....	99
Managing log files.....	99
<b>Chapter 5: MSX Licenses .....</b>	<b>100</b>
Introduction .....	100
License packs .....	101
<i>Vport-based licensing</i> .....	101
<i>License error</i> .....	101
<i>License expiration warning</i> .....	101
Obtaining an MSX license .....	101
Installing an MSX license .....	102
Preserving the license file via an MSX upgrade .....	103
Viewing license status .....	103
Version compatibility.....	104
QoS licensing .....	104
DTMF licensing.....	104
<b>Chapter 6: The MSX Database .....</b>	<b>106</b>
Database administration.....	106
<i>CLI database commands</i> .....	106
<i>Database export and import considerations</i> .....	110
<i>Manually upgrading the database</i> .....	111
<i>Replacing an existing database</i> .....	111
<b>Chapter 7: Performance Tuning and Statistics.....</b>	<b>113</b>
Configuring max calls .....	113
Patch levels required for optimization.....	113
Memory requirements.....	113
Statistics and monitoring .....	114
<i>Getting system statistics</i> .....	114
<i>Getting endpoint statistics</i> .....	115
<b>Chapter 8: SNMP Support .....</b>	<b>116</b>
Introduction .....	116
SNMP support features .....	116
<i>Net-SNMP Security Packages</i> .....	117

Configuration .....	117
<i>The SNMP configuration utility</i> .....	117
System route configuration .....	126
Controlling agent operation .....	126
Available traps .....	127
<i>SNMP Agent</i> .....	127
<i>Application</i> .....	131
Trap responses .....	136
Configuring SNMP-related options for rate limiting .....	150
<i>Configuring the SNMP trap threshold</i> .....	150
<i>Configuring TopN SNMP tables</i> .....	150
References .....	152
<b>Chapter 9: MSX Redundancy .....</b>	<b>153</b>
Introduction .....	153
<i>1+1 redundancy</i> .....	153
Dynamic endpoint registration .....	154
Call migration .....	155
Network-level redundancy architecture .....	155
<i>How to recover from a loss of control interface connectivity</i> .....	155
Database replication details .....	156
Implementing redundancy .....	157
<i>Determining and changing status</i> .....	158
<i>Implementation notes and definitions:</i> .....	159
Configuring a redundant MSX .....	160
<i>Configuring the servers' system time attributes</i> .....	161
<i>Configuring replication network interfaces</i> .....	161
<i>Configuring the control interface with nxconfig.pl</i> .....	162
<i>Configuring database replication</i> .....	163
Stateful Call Migration (SCM) .....	164
<i>Additional replicated data</i> .....	164
<i>Implementation highlights</i> .....	165
<i>Implementation requirements</i> .....	165
<i>Checking SCM status</i> .....	165
<i>Caveats</i> .....	166
<b>Chapter 10: IPSec Support .....</b>	<b>167</b>
About IPSec .....	167
How to implement IPSec .....	167

<i>Procedure for configuring IPSec on the iServer .....</i>	168
<i>Procedure for configuring IPSec on the proxy .....</i>	170
<i>How to test your IPSec implementation .....</i>	172

## PART 2:

## VOIP SERVICES - - - - -173

<b>Chapter 11: MSX Realms.....</b>	<b>174</b>
<i>NexTone's Solution.....</i>	174
Realm-based Voice over IP .....	175
<i>Endpoint registration flow .....</i>	176
<i>Outgoing call RBR flow.....</i>	176
<i>Incoming call RBR flow.....</i>	178
<i>VLAN support .....</i>	178
RBR deployment requirements .....	179
MSX RBR elements.....	179
<i>Realms and their Creation .....</i>	179
Closed realms: E.164 VPN support.....	187
CLI commands for realm-based routing .....	190
<i>Generalized realm set-up procedure .....</i>	191
<i>Output of realm list.....</i>	192
<b>Chapter 12: SIP Services.....</b>	<b>194</b>
SIP terminology .....	194
MSX as a SIP server .....	194
<i>Registration support.....</i>	195
<i>SIP over TCP support.....</i>	195
<i>SIP UPDATE support .....</i>	196
Configuring the MSX for SIP .....	207
<i>Configuring an endpoint for SIP.....</i>	207
<i>Configuring global SIP parameters.....</i>	212
<i>Other notes on SIP call processing .....</i>	217
SIP trunk group support.....	219
SIP call flow .....	220
<i>SIP NAT traversal.....</i>	222
<i>Endpoint registration.....</i>	224
SIP outbound proxy and mirror proxy .....	227

<i>Implementation highlights</i> .....	228
<i>SIP Message flows</i> .....	228
<i>Caveats</i> .....	232
<i>Related CLI commands</i> .....	233
SIP-T support .....	233
<i>ISUP</i> .....	233
<i>ANI II</i> .....	234
<i>Call Flows</i> .....	235
<i>Caveats</i> .....	236
SIP privacy .....	237
<i>Identity assertion via trusted servers</i> .....	237
<i>Untrusted endpoints</i> .....	237
<i>RFC 3325 support</i> .....	239
<i>Support for draft-ietf-sip-privacy-01.txt</i> .....	241
<i>Caveats</i> .....	243
<i>Configuring Privacy</i> .....	244
<i>MSX behavior</i> .....	244
<i>Privacy behavior</i> .....	245
<b>Chapter 13: H.323 Services</b> .....	<b>261</b>
H.323 call flows .....	261
<i>Registration and setup flow</i> .....	261
<i>Multiple routes to egress</i> .....	264
iServer as an H.323 gatekeeper .....	265
<i>RAS services</i> .....	265
<i>Routing services</i> .....	267
<i>Directory services</i> .....	269
<i>Other H.323 services</i> .....	269
H.323 protocol support .....	270
Configuring the iServer as an H.323 gateway .....	273
<i>Alternate gatekeeper support for an Sgatekeeper</i> .....	273
Information transfer capability .....	274
Configuring H.323 optional parameters .....	275
<i>Enabling H.323 on the endpoints</i> .....	275
<i>Using nxconfig.pl to configure H.323 optional parameters</i> .....	275
H.323 trunk group support .....	278
H.235 security authentication .....	279
<i>Registering to a gatekeeper using H.235</i> .....	279
<i>Configuring H.235 security authentication</i> .....	279



Logging support.....	280
Configurable local proceeding .....	280
Registrations from public and private networks.....	281
Optional H.245 address in connect message.....	282
Call progress indication .....	282
Configurable Bearer Capability (layer 1 protocol).....	282
How to configure Bearer Capability (layer 1 protocol) .....	283
Configurable Party Number Type .....	283
H.245 Capabilities Mode Restriction.....	284
How to configure H.245 Capabilities Mode Restriction.....	285
Vocaltec support considerations.....	285
<b>Chapter 14: Inter-Working Function (IWF) Services .....</b>	<b>289</b>
What is IWF? .....	289
<i>Signaling interworking and interoperability .....</i>	<i>289</i>
Supported features and protocols .....	291
Cisco Call Manager – Sonus GSX interoperability .....	291
Configuring the MSX for IWF.....	291
Forwarding signaling address for IWF Calls .....	292
Disabling g729 variants for IWF calls .....	292
IWF and DTMF Translations .....	292
<b>Chapter 15: Media Services .....</b>	<b>294</b>
Introduction .....	294
About Media Processors .....	294
<i>Inter-system communication.....</i>	<i>295</i>
<i>Media routing support.....</i>	<i>295</i>
<i>MSX Boot with Media Processor .....</i>	<i>295</i>
<i>MSX configuration .....</i>	<i>296</i>
About NSF .....	296
Setting up media devices and media routing pools .....	297
<i>Media Device Types .....</i>	<i>297</i>
<i>About the media routing pools procedures .....</i>	<i>298</i>
<i>Open the iServer configuration FCE page.....</i>	<i>300</i>
<i>Add a media routing device .....</i>	<i>300</i>
<i>Add a Vnet.....</i>	<i>301</i>
<i>Create a media resource pool .....</i>	<i>302</i>
<i>Create a media routing pool .....</i>	<i>304</i>
<i>Add a transcoder device.....</i>	<i>305</i>

<i>Create a media routing pool for transcoding</i> .....	307
Reconfiguring devices, pools, and Vnets .....	308
Deleting devices, pools, and Vnets .....	309
Implementing transcoding in MSX.....	309
<i>Preferred, prohibited, and non-preferred codecs</i> .....	311
<i>The implied non-preferred codec list</i> .....	312
<i>About endpoints negotiating codecs</i> .....	312
<i>Configuring standalone and redundant systems</i> .....	315
<i>System behavior during operation</i> .....	317
<i>System behavior during startup</i> .....	318
<i>Troubleshooting transcoder problems</i> .....	319
<i>Limitations</i> .....	319
<i>Creating codec profiles</i> .....	320
<i>Configuring endpoints and iEdge groups</i> .....	330
DTMF Translation and Tone Generation .....	331
Configuring DTMF processing .....	332
<b>Chapter 16: Firewall Control Entity (FCE) .....</b>	<b>334</b>
Supported FCE types .....	334
Configuring the MSX for FCE support .....	334
<i>Media processor support</i> .....	334
<i>NSF support</i> .....	334
<i>Configuring FCE in the servercfg table</i> .....	335
Media routing .....	335
<i>Configuring media routing</i> .....	337
Mid-Call media change (hide address change) .....	338
Preventing Rogue RTP Packets in Media Streams .....	338
<b>Chapter 17: Signaling Firewall Operations.....</b>	<b>339</b>
Firewall zones.....	339
<i>Service sets</i> .....	339
<i>Signaling vnets and realms</i> .....	340
<i>Default configuration</i> .....	340
Customizing firewall settings .....	342
<i>Firewall zone command - cli fwzone</i> .....	342
<i>Service set command - cli service-set</i> .....	343
<i>CLI options for vnets and realms</i> .....	345
<i>Enabling or disabling the signaling firewall</i> .....	345
Blacklisting sources to prevent system access .....	345

<i>Blacklisting unknown sources</i> .....	346
<b>Chapter 18: Rate Limiting</b> .....	<b>348</b>
IP layer rate limiting .....	349
<i>Defining rate limit buckets - cli ratelimitbucket</i> .....	349
<i>Assigning IP rate limits</i> .....	351
<i>Enabling/disabling IP layer rate limiting</i> .....	356
<i>Setting connection tracking timeout</i> .....	356
Signaling rate limiting at the SIP session layer .....	357
<i>Setting signaling rate limits</i> .....	357
<i>Setting reporting intervals when signaling rates exceed their limits</i> .....	360
<i>Setting the drop policy for rate limiting at the SIP session layer</i> .....	361
<i>Enabling/disabling session layer rate limiting</i> .....	362

## PART 3:

## RADIUS AAA, 3GPP Rx, AND BILLING - - - -363

<b>Chapter 19:</b> .....	<b>364</b>
<b>Chapter 19:</b> .....	<b>364</b>
Introduction .....	364
Supported services .....	364
<i>Authentication</i> .....	364
<i>Authorization</i> .....	364
<i>Accounting</i> .....	364
<b>Chapter 19: RADIUS AAA Services</b> .....	<b>364</b>
Global settings .....	365
<i>Enabling RADIUS authentication</i> .....	365
<i>RADIUS accounting</i> .....	366
<i>RADIUS database directory</i> .....	366
<i>Configuring RADIUS-MSX communication</i> .....	366
Authentication details .....	369
<i>SIP requests challenged</i> .....	370
<i>Registration flow</i> .....	370
<i>Call flows</i> .....	372
<i>Configuring SIP authentication</i> .....	374
<i>Authentication caveats</i> .....	376

Authorization details .....	376
<i>Configuring prepaid services</i> .....	377
POD details .....	377
<i>POD settings</i> .....	378
<i>POD caveats</i> .....	378
Accounting details .....	379
<i>RADIUS event sequence</i> .....	379
<i>RADIUS record layouts</i> .....	380
<i>Accounting caveats</i> .....	390
<b>Chapter 20: 3GPP Rx Interface .....</b>	<b>391</b>
Introduction .....	391
MSX Configuration Commands .....	391
Licensing .....	392
<b>Chapter 21: Billing and CDR Processing.....</b>	<b>393</b>
Introduction .....	393
NexTone-format CDRs .....	395
<i>Examples:</i> .....	405
CDR field contents.....	413
<i>ISDN cause codes</i> .....	413
<i>CDR H.225 Error Cause Code Handling</i> .....	419
<i>Called Party Types</i> .....	422
<i>QoS (Quality of Service) metrics CDR fields</i> .....	423
<i>Configuring the MSX to generate NexTone-format CDRs</i> .....	425
<i>Setting a rule for opening new CDR log files</i> .....	427
<i>Interim CDRs</i> .....	428
Hunt CDRs .....	428
CDR data transmission via RADIUS .....	429

## PART 4:

## MSX SPECIAL FEATURES - - - - -430

<b>Chapter 22: Lawful Intercept — CALEA.....</b>	<b>431</b>
Components and architecture for lawful intercept (LI).....	431
<i>The internal network interfaces for LI services</i> .....	432
Lawful intercept processing steps .....	433

How the iServer matches calls to provisioned warrants .....	433
<i>Matching rules for different target ID types</i> .....	434
Configuring LI support on iServer .....	435
Viewing LI-related information .....	440
Using RSM Console to configure lawful intercept .....	441
<b>Chapter 23: Role-based user access .....</b>	<b>443</b>
User roles .....	443
Setting up role-based user accounts .....	449
<i>Activating default users</i> .....	449
<i>Adding users to a role</i> .....	449
<i>Modifying an existing user</i> .....	450
<i>Deleting a user account</i> .....	450
iServer installation after role-based user access is implemented .....	450
<i>Rolling back after role-based user access is implemented</i> .....	451
<b>Chapter 24: VLAN Support .....</b>	<b>452</b>
VLANs .....	452
<i>VLAN high points</i> .....	452
<i>Elements</i> .....	452
<i>Signaling</i> .....	454
<i>Media</i> .....	455
<i>Element definitions</i> .....	456
<i>CLI operations</i> .....	456
<i>RSM Console operations</i> .....	458
<b>Chapter 25: Calling Plans .....</b>	<b>466</b>
What is a calling plan? .....	466
<i>Routes</i> .....	466
<i>Bulk route creation</i> .....	470
<i>Call route bindings</i> .....	470
Call legs .....	471
Creating a calling plan .....	472
<i>Prioritizing calling plans</i> .....	472
<i>Setting a time of day for calling plan operation</i> .....	473
Call trace route .....	473
Calling plan operation .....	473
<i>Destination number match operation</i> .....	475
<i>Null destination pattern</i> .....	475

<i>Destination length unspecified</i> .....	476
Calling plan application at various points .....	477
<i>Calling plan operation at ingress/source</i> .....	478
<i>Calling plan operation at egress/destination</i> .....	480
<i>Transit routes</i> .....	486
Other calling plan application scenarios .....	487
Permissive dialing.....	489
<i>Adding a route</i> .....	490
<i>Country-wide permissive dialing</i> .....	492
Basic ANI manipulation .....	494
<i>Applying ANI manipulation plans to calls</i> .....	495
File-based ANI manipulation .....	496
<i>Feature setup overview</i> .....	496
<i>Text file requirements</i> .....	496
<i>Feature example</i> .....	496
<i>Detailed setup procedure</i> .....	497
<i>Feature operation notes</i> .....	500
DNIS default route .....	501
Source port selection .....	501
<i>Source-side uports</i> .....	502
<i>Port selection algorithm</i> .....	502
<i>Example</i> .....	503
iServer trunk group support .....	503
<i>Call setup source fields</i> .....	503
<i>Trunk group parameter descriptions</i> .....	504
<i>Trunk group data pass-through options</i> .....	505
<i>Trunk group implementation and configuration</i> .....	506
<i>Practical applications</i> .....	508
<b>Chapter 26: Emergency Call Admission Control .....</b>	<b>510</b>
Overview of emergency CAC procedures .....	510
Emergency CAC .....	510
Emergency calling limits .....	511
<i>Endpoint limits</i> .....	511
<i>iEdge group limits</i> .....	511
<i>Global limits</i> .....	512
<i>Capacity consideration when setting limits</i> .....	512
Emergency number provisioning .....	512
<i>Realm-level provisioning</i> .....	512

<i>Available subcommands</i> .....	512
<i>Using the <b>list</b> subcommand</i> .....	513
<i>Using the <b>add</b> subcommand</i> .....	513
<i>Using the <b>delete</b> subcommand</i> .....	513
Procedure: Implementing emergency CAC .....	513
Related CDR field.....	514
Authentication and authorization of emergency calls .....	514
Support for interworking emergency calls .....	514
Configuring emergency calling with RSM .....	515
<i>Accessing RSM Console</i> .....	515
<i>Adding emergency numbers</i> .....	515
<i>Configuring emergency calling limits at the global level</i> .....	516
<i>Configuring emergency calling limits at the endpoint level</i> .....	516
<i>Configuring emergency calling limits at the iEdge group level</i> .....	517
<b>Chapter 27: Dynamic Call Hunting</b> .....	<b>519</b>
DCH criteria .....	519
Hunting triggers .....	520
<i>Termination entity is H.323 gateway/terminal</i> .....	520
<i>Termination entity is SIP</i> .....	521
Intra-carrier call hunting .....	522
Call hunting with multiple directory gatekeepers .....	522
<b>Chapter 28: Cause Code Operations</b> .....	<b>524</b>
Introduction .....	524
<i>Mapping</i> .....	524
<i>Hunting</i> .....	526
Factory default settings .....	526
<i>Received H.323 code mapping and hunt behavior</i> .....	527
<i>Received SIP code mapping and hunt behavior</i> .....	531
<i>Intermediate cause codes</i> .....	533
Custom configuration.....	535
<i>Configuration files</i> .....	535
<i>Creating a custom configuration</i> .....	536
<i>Setting the configuration file to use</i> .....	538
<i>Endpoint configuration</i> .....	538

**PART 5:**  
**REFERENCE APPENDICES - - - - -540**

**Appendix A: Glossary ..... A-1**

**Appendix B: The Command Line Interface ..... B-1**



## List of Figures

1.	RSM Console .....	18
2.	RSM Console MSX Restart Message .....	23
3.	Allow All Sources.....	62
4.	Entering Inoch Server Parameters .....	64
5.	MSX-to-Gatekeeper Relationships.....	66
6.	Setting Session Call Limits for an Endpoint .....	80
7.	iEdge Groups Utility Window.....	81
8.	Subscribing an Endpoint to an iEdge Group .....	82
9.	Starting the Policy Utility.....	86
10.	Add Policy Window.....	87
11.	Modify Policy Window .....	88
12.	iEdge Group Window .....	89
13.	Configuring On-net Gateways to Never Route Media .....	92
14.	Configuring an Endpoint for LNP.....	93
15.	1+1 Redundancy .....	153
16.	Implementing 1+1 Redundancy.....	157
17.	RSM Console's Control Interface Settings .....	162
18.	Realm Example .....	175
19.	Outgoing Call Signaling Example .....	177
20.	Incoming Call Signaling Example .....	178
21.	RSM Console's Add Realm Dialog Box .....	183
22.	Realm Call Routing Logic.....	188

23.	Realm Registration Logic .....	189
24.	Setting the SIP URI .....	209
25.	Enabling Broadsoft Redundancy .....	210
26.	SIP Call Setup with Multiple Addresses Returned .....	221
27.	OBP / Mirror Proxy Topology .....	228
28.	Far-end Proxy Registration .....	229
29.	Cross-realm OBP Call Setup.....	230
30.	In-realm OBP Call Setup .....	231
31.	ANI II Digit-Forwarding Route .....	235
32.	SIP endpoint configuration access .....	238
33.	SIP host trust option .....	239
34.	Example Call Flows .....	262
35.	H.323 Call Setup with Multiple Addresses Returned.....	264
36.	Local Proceeding Flow Diagram .....	281
37.	Registrations from Private and Public Networks .....	281
38.	Adding a VocalTec Gatekeeper (Phone tab).....	286
39.	Adding a VocalTec Gatekeeper (Advanced tab) .....	287
40.	Adding a VocalTec Gatekeeper (Protocol -> H.323 Configure) ..	288
41.	NexTone MSX Architecture .....	290
42.	Media Processor Interfaces.....	295
43.	iServerConfiguration FCE Page .....	298
44.	Add a New Device Dialog Box .....	300
45.	Add Media Vnet Dialog.....	301
46.	Add Vnet Media Route Dialog .....	302
47.	Add a New (resource) Pool Dialog .....	303
48.	Add a New (Media Routing) Pool Dialog .....	304

49.	FCE Page with Configured Media Processor Media Routing Pool	305
50.	Add a New Device Dialog Box .....	306
51.	Expanded Add a New Device Dialog Box .....	306
52.	Codecs in One Call Path .....	311
53.	The Transcoding Negotiation Process .....	314
54.	Transcoder configuration for a standalone MSX system.....	315
55.	Transcoder configuration for a redundant MSX system .....	316
56.	Codec Profiles Window .....	321
57.	Add Codec Profile Dialog Box .....	322
58.	Preferred Codecs List.....	323
59.	Prohibited Codecs list.....	324
60.	New Codec Profile Added .....	325
61.	Modify Codec Profile Dialog Box.....	326
62.	Codec Profiles Window .....	327
63.	EndPoint uPort with Transcoding Enabled.....	328
64.	Media Routing Interfaces .....	336
65.	iServer signaling and media .....	337
66.	Rate Limiting-basic input flow.....	348
67.	SIP Authentication Registration Flow .....	371
68.	Call Flow with RADIUS Authentication .....	372
69.	SIP Authentication for One Proxy.....	373
70.	RADIUS Authorization for Prepaid Service .....	377
71.	RADIUS Accounting, Event Sequence.....	379
72.	CDR-Related Call Flow .....	394
73.	Sample CDR Data.....	395
74.	LI architecture.....	431

75. LI network interfaces .....	432
76. Vnets Window .....	459
77. Add Signaling Vnet Dialog.....	459
78. Modify a Signaling Vnet.....	461
79. The Realms Utility (partial) .....	463
80. Assign a Signaling Vnet to a Realm .....	464
81. Assigning a Media Vnet to a Realm .....	465
82. Calling Plan Bindings .....	470
83. Sticky Routing Example .....	483
84. Adding a New Route to an Egress Gateway .....	491
85. Entering the New Area Code in an Existing Route.....	492
86. Permissive Dialing, Mexico Example .....	493
87. File-Based ANI Manipulation, Example.....	497
88. Specifying an ANI Input File .....	498
89. Binding the ANI Egress Route to the Egress Calling Plan .....	499
90. DNIS Default Route .....	501
91. Trunk Group Support in RSM Console.....	504
92. Trunk Group U-Port Example .....	506
93. Source Gateway, Sample Configuration .....	507
94. Setting ISDN CC Mapping (Partial View) .....	539

## List of Tables

1.	Typographic Conventions.....	3
2.	nxconfig.pl option parameters .....	20
3.	servercfg-set parameters .....	23
4.	Global Configuration Items.....	54
5.	Endpoint Configuration Data Items .....	67
6.	Never Route Media and Route Media Endpoint Settings, Truth Table .....	78
7.	General MSX Backup Files .....	97
8.	Net-SNMP Agent Traps.....	128
9.	NexTone Application Traps .....	132
10.	Events, Causes and Recommended Responses .....	137
11.	peering_config Block Fields .....	163
12.	Signaling Realm Attributes .....	180
13.	Realm Media Attributes .....	181
14.	Modify Realm Panel Fields.....	184
15.	Realm Media Routing Settings.....	186
16.	Commands Related to Realms .....	191
17.	Supported SIP Methods .....	196
18.	Supported SIP Responses .....	197
19.	Supported Headers that MSX can Receive or Transmit .....	205
20.	Compatibility with RFC2327 (Media Capabilities in SDP) .....	206
21.	Example SIP INVITE Trunk Group Fields .....	219
22.	Privacy Header Options .....	240

23.	ID Parameter Options.....	242
24.	SIP Origination (no privacy) to SIP Destination.....	245
25.	SIP (RFC 3325) Origination to SIP (Both) Destination, Caller ID Blocking Disabled .....	246
26.	SIP (RFC 3325) Origination to SIP (Both/ RFC 3325) Destination, Caller ID Blocking Enabled .....	247
27.	SIP (RFC 3325) Origination to SIP (Draft 01) Destination, Caller ID Blocking Disabled .....	248
28.	SIP (RFC 3325) Origination to SIP (Draft 01) Destination, Caller ID Blocking Enabled .....	249
29.	SIP (Draft 01) Origination to SIP (Draft 01) Destination, Caller ID Blocking Disabled .....	250
30.	SIP (Draft 01) Origination to SIP (Draft 01) Destination, Caller ID Blocking Enabled .....	251
31.	SIP (Draft 01) Origination to SIP (RFC 3325) Destination, Caller ID Blocking Disabled (Draft 01 to RFC 3325 conversion not supported).....	252
32.	SIP (Draft 01) Origination to SIP (RFC 3325) Destination, Caller ID Blocking Enabled (Draft 01 to RFC 3325 conversion not supported).....	253
33.	H.323 to H.323, Trust enabled .....	254
34.	H.323 to H.323, Trust disabled.....	255
35.	H.323 to SIP (RFC 3325), Trust enabled .....	256
36.	H.323 to SIP (RFC 3325), Trust disabled.....	256
37.	H.323 to SIP (Draft 01), Trust enabled .....	257
38.	H.323 to SIP (Draft 01), Trust disabled .....	257
39.	SIP (RFC 3325) to H.323, Caller ID blocking disabled.....	258
40.	SIP (RFC 3325) to H.323, Caller ID blocking enabled .....	258
41.	SIP (Draft 01) to H.323, Caller ID blocking disabled .....	259
42.	SIP (Draft 01) to H.323, Caller ID blocking enabled .....	260

43.	H.323 Protocol Support .....	270
44.	Information Transfer Capability Options .....	274
45.	IWF Feature Support .....	291
46.	IWF Protocol Support .....	291
47.	DTMF Translation Matrix .....	293
48.	System behavior when a transcoder failure is detected during operation 318	
49.	System behavior when transcoder cannot be reached during startup	318
50.	CLI commands used to configure endpoints and iEdge groups for transcoding .....	330
51.	DTMF Translation Matrix .....	331
52.	fwzone command .....	342
53.	service-set command .....	343
54.	blacklist command .....	346
55.	ratelimitbucket command .....	349
56.	RADIUS Parameters .....	369
57.	sipauth CLI Command Syntax .....	375
58.	RADIUS XA3+ Record Fields .....	381
59.	RADIUS XA3+ Acct-Session-Id Sub-Fields .....	383
12.	RADIUS 12.2(11)T Record Fields .....	385
37.	Release-Source Value Descriptions .....	389
13.	3GPP Initial Configuration Parameters .....	392
14.	Normal CDR File Suffixes by Type .....	395
15.	General CDR Fields .....	396
16.	Example of CDR Fields .....	405
17.	Call Disconnect CDR Field (field 13) .....	408
18.	Error Type CDR Field (field 14) .....	409

19.	Listed ISDN Cause Codes (field #30) .....	413
20.	Supported RAS Reason Codes (field #43).....	419
21.	Supported H.225 Error Codes (field #44).....	421
22.	Called Party Types (fields 51 and 52) .....	422
23.	Warrant matching examples.....	434
24.	mDevices parameters for LI .....	437
25.	Servercfg parameters for SS8 DF servers .....	438
26.	CLI commands to monitor LI processing .....	440
27.	Privileges and commands available for each role .....	444
28.	Commands allowed for nxuser.....	446
29.	Commands allowed for nxintercept .....	448
30.	cli Vnet Commands for Signaling Vnets .....	457
31.	cli realm edit Command for Signaling Vnets .....	458
32.	Route-Defining Parameters .....	468
33.	Example Text File Field Definitions .....	494
34.	Trunk Group Circuit ID Data Pass-Through .....	506
35.	H.323 Factory Configuration .....	528
36.	SIP Factory Configuration .....	532
37.	Intermediate Codes, Factory Configuration.....	534
B-1.	iServer Commands.....	B-2



## About This Guide

The NexTone MSX provides a SIP/H.323 interworking element and Firewall Control Entity (FCE) on a single platform that is deployed at your network edge or core. This guide contains elements of theory and step-by-step procedures that show you how to configure and manage the MSX platform.

### Who Should Use This Guide

This document is written for Network Engineers, System Administrators, NexTone-Certified Engineers, NexTone Field Engineers, and NexTone Technical Consultants who have been tasked with setting up, managing, and troubleshooting the MSX. This guide assumes that those who use it will have a basic knowledge of the NexTone solution and of VoIP protocols, as well as a solid working knowledge of Linux.

Before using this guide to configure your MSX, read the NexTone MSX Installation Guide.

### What's in this guide

Content in this guide consists of the following chapters and appendices organized by part:

- ◆ **Part 1: MSX System**
  - Chapter 1: About This Guide
  - Chapter 2: What Does the MSX Do?
  - Chapter 3: MSX Configuration
  - Chapter 4: MSX Administration
  - Chapter 5: MSX Licenses
  - Chapter 6: MSX Database
  - Chapter 7: Performance Tuning and Statistics
  - Chapter 8: SNMP Support
  - Chapter 9: MSX Redundancy

- Chapter 10: IPSec Support
- ◆ **Part 2: VoIP Services**
  - Chapter 11: MSX Realms
  - Chapter 12: SIP Services
  - Chapter 13: H.323 Services
  - Chapter 14: Inter-Working Function (IWF) Services
  - Chapter 15: Media Services
  - Chapter 16: Firewall Control Entity (FCE)
- ◆ **Part 3: RADIUS, 3GPP, and Billing**
  - Chapter 17: Signaling Firewall Operations
  - Chapter 18: Rate Limiting
  - Chapter 19: Radius AAA Services
  - Chapter 20: 3GPP Rx Interface
  - Chapter 21: Billing and CDR Processing
- ◆ **Part 4: MSX Special Features**
  - Chapter 22: Lawful Intercept
  - Chapter 23: Role-based User Access
  - Chapter 24: VLAN Support
  - Chapter 25: Calling Plans
  - Chapter 26: Emergency Call Admission Control
  - Chapter 27: Dynamic Call Hunting
  - Chapter 28: Cause Code Operations
- ◆ **Reference Appendices**
  - Appendix A: Glossary
  - Appendix B: The Command Line Interface

## Typographic conventions

For consistency, the conventions described in Table 1 have been used in this Guide.

**Table 1. Typographic Conventions**

This convention...	Stands for...
Names of GUI fields in which you can enter values are given in plain Helvetica type.	Enter the address of the host machine in the IP Address field
CLI commands are set in <code>Courier New</code> face.	Type this command at the prompt: <code>iserver all start</code>
Information that you must supply is shown with an <u>underline</u> , as with standard <code>man-page</code> format.	Create a directory for the MSX: <code>mkdir <u>directory name</u></code>
Hold down the Ctrl key, then press S.	Press <Ctrl>+S
Terminal window excerpts and human-readable text file contents are shown in monospaced <code>Courier</code> type.	<CR_SRC> " " </CR_SRC>

## What Documentation Is Required

### **NexTone Documentation**

- ◆ *Real-time Session Manager (RSM) Operations Guide*, release 4.3
- ◆ *Multiprotocol Session Exchange (MSX) Installation Guide*, release 4.3
- ◆ *NexTone iServer (MSX) Feature Guide for Softbank*, release
- ◆ *Real-time Session Manager (RSM) Release Notes*, release 4.3

### **Related Documentation**

- ◆ *IETF Request for Comments (RFC) references can be found at <http://ietf.org/rfc.html> or at <http://www.faqs.org/rfcs>*
- ◆ *H.323: Packet-based multimedia communications systems, ITU-T [www.itu.int](http://www.itu.int)*

◆ *SIP: Session Initiation Protocol, IETF tools.ietf.org*

## How to Contact NexTone Support

If you need to contact NexTone's Customer Support, do the following:

1. Go to NexTone's web site at: <http://www.nextone.com/>.
2. Select **Support** from the main menu on the NexTone home page. A description of available support options is displayed after login.
3. For urgent issues, you are encouraged to call our Support Hotline at +1 (240)686-3983 for immediate assistance.

## What Does the MSX Do?

---

NexTone products enable circuit-switched and VoIP networks to interconnect simply and cost-effectively. NexTone MSX provides a SIP/H.323 interworking element and Firewall Control Entity (FCE) on a single platform that is deployed at the network edge or core.

The MSX iServer is the software package that enables the VoIP application capability. Both the iServer software and the MSX session border controller hardware make up the entire MSX product package.

The MSX provides the following session border controller capabilities:

- A SIP proxy server (stateless or stateful)
- An H.323 Gatekeeper
- A SIP/H.323 InterWorking Function (IWF)
- Firewall control for both SIP and H.323 calls (FCE)
- Transcoding device support
- Multi-vendor interoperability
- Policy and routing to handle call origination and termination

### Who should use this document

The primary intended audience for this guide is those persons involved with deploying, implementing, configuring, and maintaining a NexTone MSX-based solution. This includes:

- Network Engineers
- System Administrators
- NexTone Field Engineers and Technical Consultants
- NexTone Certified Engineers

This document details the procedures required to install and operate a NexTone MSX. It is meant for use primarily by system administrators.

## MSX components and services

The MSX system supports the two major VoIP protocols: H.323 and SIP. This includes gatekeeper (H.323) and SIP proxy. An H.323-to-SIP interworking function (IWF) is also provided, as is firewall control (via FCE).

### **MSX configuration data**

The iServer software uses a set of global parameters the control error logs, Call Detail Record (CDR) output, and other global system settings. Global information is set using the `nxconfig.pl` utility described in MSX Configuration, on page 10, though typically under the direction of NexTone Customer Support. Typical iServer operations are executed through RSM Lite or RSM Console.

### **H.323 gatekeeper services**

In an H.323 network, the MSX system is compliant with the Version 4<sup>1</sup> suite (ITU-T H.323 suite) of standards, and can operate:

- As an H.323 gatekeeper
- As a domain controller

By default, the MSX is configured as an H.323 gatekeeper, and can be used for:

- Registration and Admission and Status (RAS) services
- Routing services
- Directory services
- Other services such as proxy as gateway, multiple domain support, interoperability with other gatekeepers and CDR support.
- Security

### **Inter-Working Function (IWF) services**

The MSX acts as a bridge between multiple hybrid networks, facilitating calls between H.323 and SIP endpoints. When routing H.323 calls, the MSX acts as a gateway and gatekeeper; when routing SIP calls, it acts as a user agent. When using this feature to route calls, the MSX can generate Call Detail Records (CDRs) for all calls, irrespective of the protocol used.

---

1. The MSX release covered in this guide incorporates a Version 4-compliant H.323 protocol stack, which also supports versions 2 and 3 of the H.323 standard.

### **Firewall Control Entity (FCE) services**

The MSX can act as a stateful Firewall Control Entity (FCE), ensuring network security. When configured for this feature, the MSX provides dynamic Network Address Translation (NAT) and access control services for the network as well.

### **NexTone Session Filter (NSF) services**

The NexTone Session Filter (NSF) is a packet filter/forwarder that allows RTP media streams to flow through the MSX. Unlike NSF-NP, NSF uses no special hardware and generic Intel gigabit Ethernet ports. NSF can be run on Westville Enterprise, Westville NEBS (Langley-Prestonia) platforms, and is compatible with MSX versions 4.2 and higher in all media routing functions. NSF cannot be loaded on systems that use NSF-NP. NSF provide two dedicated Ethernet ports for media streaming, in addition to dedicated ports for management, redundancy control, and signaling. Extra Network Interface Cards (NICs) must exist for those systems that do not have sufficient ports.

### **NexTone NSF-NP Network Processor Board**

The NSF-NP is a network processor board that increases media-routed call capacity and performance. For a detailed explanation of NSF-NP, see *Chapter 15, Media Services*.

## **Tools for configuring MSX**

You can configure and manage the MSX using the following tools:

- RSM Console and RSM Lite™:
  - RSM Console is available by clicking a link on the RSM Web page for installations that include full RSM.
  - RSM Lite replaces the iView stand-alone program supplied for previous releases. Access to it is Web browser-based, but the executable loads on the fly directly from the MSX. RSM Lite is primarily intended for installations not having the full RSM.
- The SOAP-based WebServices Application Programming Interface (API). See the *RSM WebServices (API) User Guide* for details and supported WSDLs (web services description language).
- RSM, the NexTone Real-time Session Manager. Provides additional functionality, such as partitioning, to the MSX. RSM provides much functionality of its own, like CDR-based reporting and alarming, and least-cost routing (LCR).

- Command Line Interface (CLI). Refer to Appendix B for information on using the CLI.

## Real-time Session Manager (RSM) Components

### ***Real-time Session Manager (RSM)***

The NexTone Real-time Session Manager (RSM) allows one user to monitor and control multiple MSXs from a single access point and application. Through RSM, a user can run the RSM Console, from which to manage their MSXs. RSM also provides much additional functionality, such as service status monitoring, performance, event handling, CDR streaming, and revenue and other business reports. Contact your NexTone representative for more information on RSM. RSM operations and installation guides are available from NexTone.

### ***RSM Lite™ and RSM Console™***

The NexTone RSM Lite provisioning and configuration application provides a user-friendly graphical user interface (GUI) for most network provisioning tasks. RSM Lite provides a familiar windowing interface for entering information for most tasks that can be performed with the NexTone MSX. RSM Lite is launched from a web browser, and runs from the MSX platform.

RSM Console provides similar functionality to RSM Lite, but is tailored to systems that are combined with RSM. For example, RSM Console has the ability to set up and control partitions, because an MSX system controlled by RSM has partitioning capability. RSM Console is launched from RSM's Web page.

Throughout this guide, you will see references to RSM Console when speaking of GUI-based operations. Unless specifically stated otherwise, all references to RSM Console also apply to RSM Lite. These replace the stand-alone iView product used with prior versions.

RSM Console and RSMLite both include online help facilities.



# **Part 1:**

## **MSX System**

## MSX Configuration

For the NexTone MSX to operate in a network, you must first add endpoints to the MSX's database. Once this information has been entered, the endpoints can dynamically register with the MSX, using the registration ID (*regid*.) under which you entered that endpoint into the database. The MSX routes calls based on the endpoint configuration information stored in its database.

Note that to support device mobility, an endpoint that's already registered can send a new registration from a different IP address. The new IP address will replace one that's already in the database for that *regid*. Two endpoints should never share a *regid*, since both cannot be simultaneously registered, because the second will overwrite the IP address for the first, and if one unregisters, the other has its status effectively changed to "unregistered" also.

Alternately, you can configure "static endpoints" directly on the MSX. Static endpoints do not have to register with the MSX, which considers them "active" at all times.

You can configure both static and registered endpoints on the MSX using RSM Console NexTone's GUI-based provisioning and configuration application. RSM Console comes with detailed online help that lists the steps to configure endpoints on the MSX.

*Note: The standalone iView client is no longer available. Use the RSM Console as described in Using RSM console on page 16.*

### Supported endpoint types

The MSX supports the endpoint types listed below. The subsections that follow describe each type.

- Generic IP device
- H.323 gateway
- H.323 gatekeeper
- Master gatekeeper, also called the "super gatekeeper" or the Sgatekeeper
- SIP gateway

- SIP proxy
- Softswitch
- User account
- ENUM server

For each type of configured endpoint, the MSX uses a different set of parameters. The particulars for each type are listed in the sections below. The parameters are described (with their `cli` equivalent keywords) in Table 5 on page 67.

### **Generic IP device**

When you configure a generic IP device on the MSX session controller, you can control all the options (such as enabling H.323 or SIP) on the device. Using RSM Console, you can configure the parameters shown below for a generic IP device endpoint on the session controller.

#### **Minimum required fields**

A generic IP device, such as an IP phone, requires that the following fields be set. The other applicable fields are optional.

- Device type of Generic IP Device (`ipphone` in `cli`)
- Registration ID
- Port number (0-255)
- Extension / phone number
- Realm in which the endpoint resides
- SIP or H.323 protocol support enabled
- NAT detection enabled, if the endpoint is behind a NAT device

Based on the information you entered for this endpoint, the MSX deduces the following information about the endpoint:

- Phone number
- VPN phone number
- VPN name
- VPN group

### **H.323 gateway, gatekeeper & sgatekeeper**

Configuring an endpoint as an H.323 gateway automatically disables SIP on that endpoint. Using RSM, you can configure in the MSX the same parameters

for a non-static H.323 gateway, as you can for a generic IP device, except for SIP-specific parameters.

Note that when assigning a calling plan to an MSX operating as a gateway, you may specify its subnet IP in place of the IP address of the source device. The MSX uses this subnet IP to deduce the source network or endpoint, to apply the calling plan to the source. A detailed description of this is given in *Calling plan operation at ingress/source* on page 478.

### **About the sgatekeeper**

While the MSX can operate as a gatekeeper, some vendors' gatekeepers (Clarent, for example) are not made to intercommunicate with gatekeepers other than their own. To address this situation in a mixed environment (e.g., NexTone gatekeepers and Clarent gatekeepers working together), the MSX is configurable to appear to the other gatekeeper as if the MSX is a gateway, not a gatekeeper. To do this, you configure the other vendor's gatekeeper on the MSX as a "super gatekeeper," or Sgatekeeper.

### **H.323 gateway, minimum required fields**

An H.323 gateway requires the fields below. Other applicable fields are optional.

- Device type of H.323 Gateway (known as `xgateway` in `cli`)
- Registration ID
- Port number (0-255)
- IP address
- Realm the gateway or gatekeeper resides in
- If an egress gateway, a calling plan

### **H.323 gatekeeper, minimum required fields**

An H.323 gatekeeper requires the fields below. Other applicable fields are optional.

- Device type of H.323 Gatekeeper (`xgatekeeper` in `cli`)
- Registration ID
- Port number (0-255)
- IP address
- Realm the gateway or gatekeeper resides in
- If an egress gateway, a calling plan

**H.323 super (master) gatekeeper, minimum required fields**

An H.323 sgatekeeper requires the fields below. Other applicable fields are optional.

- Device type of Master Gatekeeper (sgatekeeper in cli)
- Registration ID
- Port number (0-255)
- IP address
- MSX defined default realm
- If an egress gateway, a calling plan
- Tech Prefix, H.323 Id or Gatekeeper ID may be required for an MSX to register with this device

*Note: For the sgatekeeper functionality to work, there must be a default realm, and the sgatekeeper must be assigned to it. Create the default realm if one does not already exist, using the command below. Then assign the sgatekeeper to that realm.*

```
cli realm edit realm name default enable
```

**SIP gateway/SIP proxy**

Configuring an endpoint as a SIP gateway or SIP proxy automatically disables H.323 on that endpoint. Using RSM Console, you can configure in the MSX the same parameters for a non-static SIP gateway or SIP proxy endpoint, as you can for a generic IP device, except for H.323-specific parameters.

**SIP gateway, minimum required fields**

A SIP gateway requires the fields below. Other applicable fields are optional.

- Device type of SIP Gateway (sipgw in cli)
- Registration ID
- Port number (0-255)
- IP address
- Realm the gateway or gatekeeper resides in
- If an egress gateway, a calling plan

**SIP proxy, minimum required fields**

A SIP gateway requires the fields below. Other applicable fields are optional.

- Device type of SIP Proxy (sipproxy in cli)
- Registration ID

- Port number (0-255)
- IP address
- Realm the gateway or gatekeeper resides in
- If an egress gateway, a calling plan

### **Softswitch/ENUM server/MSX**

When you configure an endpoint as a softswitch, ENUM server or MSX, both H.323 and SIP are automatically enabled on that endpoint. Using RSM Console, you can configure in the MSX the same parameters for a non-static softswitch, ENUM server or MSX endpoint, as for a generic IP device (except for call forwarding data).

### **ENUM server**

An *ENUM* server converts E.164 (telephone) numbers into Internet URIs, using DNS lookups. An MSX supports multiple defined ENUM Server endpoints, with the limitations described in *Multiple ENUM server limitations*, below.

To configure ENUM services:

1. Set the global ENUM domain:
  - Using the `nxconfig.pl` command:
 

```
nxconfig.pl -e enumdomain -v domain name
```
  - In RSM Console, right-click the server and choose **Configure**, then click the **System** tab, followed by the **Other** tab. Enter the value in the ENUM domain box.
2. Provision an ENUM server as an endpoint of type ENUM:
  - In CLI, use:
 

```
cli iedge edit regid uport type enum
```
  - In RSM Console, locate the device and select **Device Type** of **ENUM Server**

The MSX supports multiple ENUM server endpoints. You select the setting on the **Phone** tab for the device, through the **Device Type** pull-down menu.

3. Assign an ENUM domain to the endpoint you named in step 2, using the CLI command:

```
cli iedge edit regid uport enumdomain domain string
```

4. Add to the NexTone MSX database a Domain Name Server (DNS) that has in its address table the ENUM server you provisioned in step 2. When added this way, the MSX sends all queries as domain name queries to the DNS server.

**Note:** *The MSX uses the trial.e164.com domain by default for all ENUM domain name resolutions. If the DNS referred to above is different from this domain, you MUST add it to the MSX either using RSM Console (see RSM Console's online help) or with the nxonfig.pl utility (see Setting the global default ENUM domain on page 50).*

#### **Multiple ENUM server limitations**

- The MSX cannot support multiple ENUM servers with the same private IP address, even if they are a part of different realms.
- The MSX does not support a configuration where more than one ENUM server is configured with the same domain for redundancy purposes.
- ENUM domains can be assigned to ENUM servers through `cli` commands, but not RSM Console or Lite.

#### **User account**

If you wish to only configure phone numbers to indicate your presence at different times/places, you can configure a user account endpoint on the MSX instead of configuring a device. Both H.323 and SIP are automatically disabled on this type of endpoint. Using RSM Console, you can configure the following parameters for an MSX user account endpoint:

#### **User account, minimum required fields**

A user account requires the fields below. Other applicable fields are optional.

- Device type of User Account (`user` in `cli`)
- Registration ID
- Extension
- Realm the user resides in

#### **Inoch server**

An Inoch server is supported for performing local number portability (LNP) dips. Parameters needed for the MSX to establish communication with the Inoch server include:

- The Inoch server's IP address
- The Inoch server port to which the MSX will send LNP dip requests

- The timeout interval for the MSX to wait before concluding that the Inoch server is unavailable.

See *Local Number Portability (LNP) support* on page 63 for the Inoch server configuration procedure.

## MSX global settings

Global configuration settings for the MSX are stored in a table of values named *servercfg*. For information on setting global parameters, see *Global configuration using nxconfig.pl* on page 18.

*servercfg* settings can be edited using either the RSM Console/Lite or the *nxconfig.pl* configuration utility, according to your preference. These two MSX configuration interfaces are described in *Understanding MSX configuration interfaces* on page 16.

Changes to *servercfg* settings are automatically replicated to secondary MSXs in redundant MSX configurations, with settings specific to a given MSX in a redundant configuration transparently maintained.

## Understanding MSX configuration interfaces

There are four main interfaces for configuring the MSX software:

- RSM Console (or RSM Lite)
- the *nxconfig.pl* global parameter configuration utility
- the *cli* command-line utility.
- RSM provides a web interface for managing some advanced aspects of your MSX, like report and alarm management, and least-cost routing (LCR).  
RSM is available as a separate product with its own product documentation.

The *nxconfig.pl* utility is introduced in *Global configuration using nxconfig.pl* on page 18. The *cli* command line utility provides an interface to most of the data tables used by the MSX software. Documentation for CLI commands are scattered throughout this book, and summarized in Appendix B, “The Command Line Interface”.

The RSM Console, and instructions for launching it, are provided below.

### Using RSM console

RSM Console is a graphical user interface to the MSX data tables. It is launched from one of the following web applications:



- RSM, installed on an RSM workstation configured to manage your MSX
- RSM Lite, installed on the MSX

See your RSM IOG for information about launching the RSM Console interface from an RSM workstation. If you do not have an RSM workstation, launch the RSM Console from the RSM Lite web application server, which is installed on your MSX.

To start RSM Console from RSM Lite:

1. Open a web browser on a Windows or X-Windows workstation and in the Address field, enter:

`https://iserver mgmt ip/rsm`

where `iserver mgmt ip` is the IP address of the MSX's management interface. The RSM Lite Login screen appears.

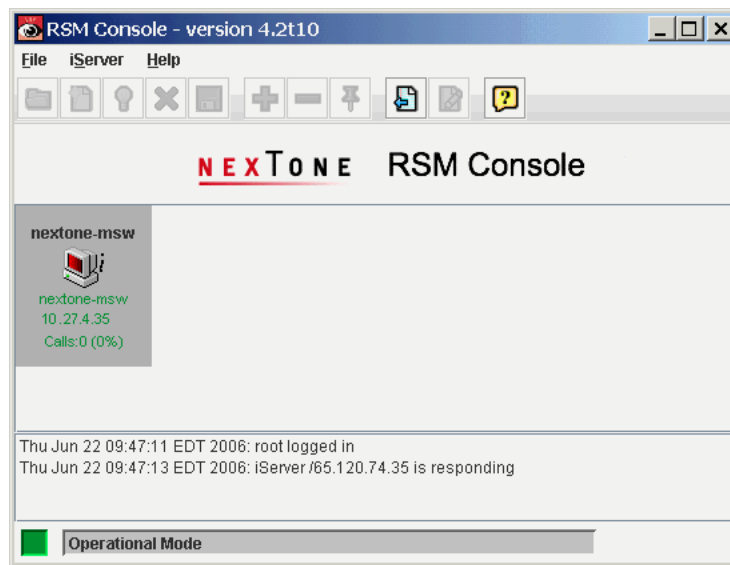
2. Enter the default RSM Lite username and password. If you do not know the default username and password, contact NexTone Support.

A web page with two links—RSM Console and Logout—in the upper left corner appears.

- Click the RSM Console link. If this is the first time you have used RSM Console, it is downloaded to your workstation and then launched. The initial RSM Console window appears in the following figure.

*Note: See the RSM Console online help for information about using RSM Console, and the task-oriented descriptions in this document for information about performing configuration tasks with RSM Console.*

**Figure 1. RSM Console**



## Global configuration using nxconfig.pl

You can change the configuration values stored in *servercfg* using the *nxconfig.pl* utility. Values settable this way fall into these categories:

- Billing data (including RADIUS on/off, if RADIUS is licensed), and Call Detail Records (CDR) options
- Other MSX-specific parameters, such as passwords and configuration server parameters
- SIP options
- H.323 options
- Other options, such as *ENUM domain* and *allow all source numbers*
- Firewall Control Entity (FCE) parameters
- MSX peering for redundancy, database replication, etc.

### ***Running the nxconfig.pl utility***

There are two ways to set *servercfg* parameters with `nxconfig.pl`: Individually or interactively. Both require you to log in to the MSX, either at the console or via `ssh`.

#### **nxconfig.pl syntax**

The `nxconfig.pl` utility has the following syntax:

```
nxconfig.pl [options] [-D dbname -u dbuser -h dbhostip -p dbpasswd]
```

The `-D`, `-u`, `-h`, and `-p` parameters are not normally required. The `nxconfig.pl` options determine the action the script performs. These options are described in Table 2 on page 20.

**Table 2. nxconfig.pl option parameters**

Options	Description
-S	Lists all attribute name-value pairs in <i>servercfg</i> .
-s <u>attribname</u>	Displays formatted, detailed information about the specified attribute, including its process, category, name, default value, data type, description text, and whether a process restart is required when the attribute is changed.
-E	Used to edit a subset of <i>servercfg</i> values. User is prompted for values for all attributes in the subset.
-e <u>attribname</u> [-v <u>attribvalue</u> ]	Edits a single attribute. The user is prompted for a value if -v <u>attribvalue</u> is not provided.
-M	Writes contents of the <i>mdevices.xml</i> attribute to a new <i>mdevices.xml</i> file in the current directory.
-m -P [ <u>path</u> ]	Updates the <i>mdevices.xml</i> attribute with contents of file <i>&lt;path&gt;/mdevices.xml</i> (if -P is specified) or <i>./mdevices.xml</i> (if -P is not specified). This updates the system <i>mdevices</i> data.  <i>Note: This should only be used under the direction of NexTone Support.</i>
-d	Used to edit the <i>mdevices.xml</i> attribute. This option opens the default editor ( <i>vi</i> is the MSX's default editor) with the contents of the <i>mdevices.xml</i> attribute. You can edit and save the XML in the editor to write changes back to <i>servercfg</i> .  <i>Note: This should only be used under the direction of NexTone Support.</i>
-L	Retrieves the content of the <i>iserverlc.xml</i> attribute and writes it to file <i>iserverlc.xml</i> in the current directory.
-l [-P <u>path</u> ]	Updates the <i>iserverlc.xml</i> attribute with contents of file <i>&lt;path&gt;/iserverlc.xml</i> (if -P is specified) or <i>./iserverlc.xml</i> (if -P is not specified). This updates the system license.

**Setting individual *servercfg* parameters:**

The names of the attributes you can individually set with the *nxconfig.pl* utility are listed in Table 3, beginning on page 23.

To set the value of an individual *servercfg* attribute:

1. Set the attribute value. At the command prompt, enter:

```
nxconfig.pl -e name -v value
```

where name is the name of a user-configurable attribute and value is the value you are assigning to it. If you don't enter -v value, you will be prompted for a value for the specified name.

**Note:** Always put double quotation marks around non-integer (string) values. For example, to assign the value "bar" to the attribute foo, enter:

```
nxconfig.pl -e foo -v "bar"
```

2. Optionally, stop and restart the MSX. Changes to some attributes require that you restart the MSX software in order to make the change effective. When this is required, nxconfig.pl displays:

```
Notice: iServer restart required! Restart now (y/n) [n]:
```

Type y to restart the MSX processes. Note that in-process H.323 calls, all incomplete call setups, are dropped during a restart.

### **Manually re-starting the MSX processes**

To stop and restart the MSX manually, enter the following commands, in the order shown:

```
iserver all stop
iserver all start
```

### **Setting servercfg parameters interactively:**

The nxconfig.pl utility supports an interactive mode, that prompts for a subset of editable values. The current value of each attribute is shown. To cycle through the prompts, either enter a value at a prompt, then press <Enter>, or simply press <Enter> to accept the current value. To exit the program and register your changes, press <Enter> repeatedly until all prompts have appeared. To exit the utility at any time, discarding any changes you may have made, press <Ctrl>+C. Changes to some parameters require an MSX process restart. These are noted in Table 3 on page 23.

**Note:** Enter a "?" at any nxconfig.pl prompt to see help for the prompt.

To run nxconfig.pl in interactive mode:

1. At the command prompt, enter:

```
nxconfig.pl -E
```

2. As each attribute appears, enter a new value, or press <Enter> to keep the current setting. When all prompts have appeared, putting them into effect is a two-step process, as follows.
  3. First, your responses are shown for the prompted attributes, followed by:  
Do you want to commit the changes (y/n)? [y]
  4. Press <Enter> to save (commit) your changes, or type n and press <Enter> to discard them. Changes you save, that do not require an MSX restart go into effect immediately.
  5. If you save your changes, and have changed a parameter that requires a restart, a second prompt appears:  
Changes made require iServer restart (y/n) [y]
- Note:** In-process H.323 calls, all incomplete call setups, are dropped during a restart.
6. Press <Enter> to restart the MSX servers affected by the prompted values, or type n and press <Enter> to skip the MSX restart. Note that if you enter an n, values you changed that require a restart do not go into effect until the MSX(s) are restarted the next time. See *Manually re-starting the MSX processes* on page 21 to perform a manual restart of MSX processes. Note that restarting these processes will cause in-progress call setups to be dropped.

### **servercfg attributes**

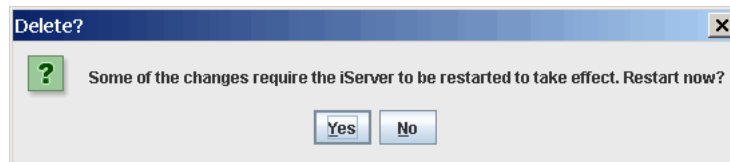
Table 3 lists all of the attributes in the *servercfg* area. All of these attributes can be changed using the *nxconfig.pl* utility, and many of them can also be edited using RSM Console.

#### **About servercfg attributes that require an MSX restart**

Many of the attributes should only be changed under the direction of NexTone Support. In addition, some attributes require an MSX process restart to put changes to them into effect. These attributes are noted in the descriptions in Table 3.

When you use RSM Console to change an attribute that requires a restart, this message appears:

**Figure 2. RSM Console MSX Restart Message**



Click **Yes** to restart immediately, or **No** to defer the restart until later. To perform a manual MSX restart at any time, log onto the MSX and perform the steps in *Manually re-starting the MSX processes* on page 21.

**Table 3. servercfg-set parameters**

Attribute Name	Description	Default
acctsessionid-overloaded	Select the XA3+ format for RADIUS communication. Valid values: 1 (Enable)   0 (Disable). Requires MSX restart.	0
age-timeout	The time interval for which a registered endpoint is considered active, before which it must re-register. The default value is 900 seconds. The actual working value is based on a number of other factors.	900
allow-ani-portssel	Allow port selection based on ingress ANI if allow all sources is enabled. Valid values: 1 (Enable)   0 (Disable)	0
allow-autharq-all	Authenticate all originating H.323 endpoints with the master gatekeeper. Valid values: 1 (Enable)   0 (Disable)	0
allow-dest-all	Valid values: 1 (Enable)   0 (Disable)	0
allow-destarq-all	Respond to an ARQ from a gateway with an ACF and allow call to go through even though the iServer cannot control the session. Valid values: 1 (Enable)   0 (Disable)	0

Attribute Name	Description	Default
allow-dnis-portsel	Allow port selection based on ingress DNIS if allow all sources is enabled. Valid values: 1 (Enable)   0 (Disable)	0
allow-dynamicendpoints	Allow dynamic endpoint registration in OBP mode. Valid values: 1 (Enable)   0 (Disable)	0
allow-dynamicendpointslss3263	This will do DNS lookup according to RFC3263 for locating dynamic endpoints using NAPTR, SRV and A records if needed. Valid values: 1 (Enable)   0 (Disable)	0
allow-dynamicendpointslsssrv	This will skip the NAPTR query while doing a DNS lookup for locating dynamic endpoints. Valid values: 1 (Enable)   0 (Disable)	0
allow-src-all	Allow MSX to accept incoming calls from all sources. Valid values: 1 (Enable)   0 (Disable)	0
altgklist	List of alternate gatekeepers in the format IP:port,IP:port,IP:port,...	[Null]
always2833	In a SIP-H323 call, send RFC2833 dynamic RTP payload type 101 in the 200Ok to the originating SIP endpoint. Valid values: 1 (Enable)   0 (Disable)	0
billingtype	Billing type options: none   postpaid   ciscoprepaid   prepaid.	postpaid
callidinlrq	Insert the call ID in an LRQ sent to a peer gatekeeper. Valid values: 1 (Enable)   0 (Disable)	
cdrcallidlen	Print at the most these many characters of a Call ID in the CDRs.	64
cdrrerrorstopcalls	Stop processing calls if CDRs cannot be written due to disk full. Valid values: 1 (Enable)   0 (Disable)	0



Attribute Name	Description	Default
cdrevents	Space-separated list of type of CDRs written. CCR events are start1, start2, end1, end2 and hunt. By default, end1 CDRs will always be written and that cannot be changed through this config. If you also want to write start1, end2 and hunt CDRs, this field would contain: start1 end2 hunt	end1
cdrformat	CDR format: mindcti, xml or syslog. Default is mindcti.	mindcti
cdrtimer	Time interval in minutes for time based CDRs	60
cdrtype	CDR type options: fixed   daily   time   seq. CDRs can be written at a fixed file size   on a daily basis   at a specified time interval   sequentially.	daily
control-interface	Control interface used to communicate with peer servers. Making a change to this requires an MSX restart.	[Null]
daemonize	This decides whether the gis is started up in daemonized mode or not. This parameter is pre-configured by NexTone and should not be changed. Valid values: 1 (Enable)   0 (Disable). Making a change to this requires an MSX restart.	1
debug-modage	Log level for module <b>age</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modares	Log level for module <b>ares</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modarq	Log level for module <b>arq</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0

Attribute Name	Description	Default
debug-modbridge	Log level for module <b>bridge</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modcache	Log level for module <b>cache</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modcdr	Log level for module <b>cdr</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modcli	Log level for module <b>cli</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modconn	Log level for module <b>conn</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-moddb	Log level for module <b>db</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-moddbsync	Log level for process <b>dbsync</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-moddef	Log level for module <b>def</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-moddlic	Log level for module <b>dlic</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modevt	Log level for module <b>evt</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modexecd	Log level for process <b>execd</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0

Attribute Name	Description	Default
debug-modfaxp	Log level for module <b>faxp</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modfce	Log level for module <b>fce</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modfind	Log level for module <b>find</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modh323	Log level for module <b>h323</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modicmp	Log level for module <b>icm</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modinit	Log level for module <b>init</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modinoch	Log level for module <b>inoch</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modirq	Log level for module <b>irq</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modispd	Log level for module <b>ispd</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modiwf	Log level for module <b>iwf</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modlmgr	Log level for module <b>lmgr</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0

Attribute Name	Description	Default
debug-modlrq	Log level for module <b>lrq</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modpkt	Log level for module <b>pkt</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modpmgr	Log level for process <b>pmgr</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4). Making a change to this requires an MSX restart.	0
debug-modq931	Log level for module <b>q931</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modquedb	Log level for module <b>quedb</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modradc	Log level for module <b>radc</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modredund	Log level for module <b>redund</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modreg	Log level for module <b>reg</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modrrq	Log level for module <b>rrq</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modscc	Log level for module <b>scc</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modscm	Log level for module <b>scm</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0

Attribute Name	Description	Default
debug-modscmrpc	Log level for module <b>scmrpc</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modsel	Log level for module <b>sel</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modshm	Log level for module <b>shm</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modsip	Log level for module <b>sip</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modtcf	Log level for module <b>tcf</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modtmr	Log level for module <b>timer</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
debug-modxml	Log level for module <b>xml</b> . Levels of logging available are: warn(1), error(2), debug(3), trace(4)	0
default-codec	Default MSX codec options: g711ulaw64k,g711alaw64k,g729 or g723.1. If MSX receives a slow-start Setup or Invite without SDP & the terminating endpoint is a SIP endpoint, the MSX sends it an Invite with SDP containing the default codec specified here	g711ulaw64k
defaulttts2833	Send RFC2833 capability with dynamic RTP payload type 101 in default TCS. Valid values: 1 (Enable)   0 (Disable)	0
dnsrecoverytimeout	Timeout for DNS query in seconds	120

Attribute Name	Description	Default
enable-autounregister	If a dynamic endpoint fails to meet the MSX shorter keep-alive interval, the MSX actively de-registers the endpoint from the remote registrar. Valid values: 1 (Enable)   0 (Disable)	0
enable-natdetection	Allow detection of dynamic endpoints behind a NAT. Valid values: 1 (Enable)   0 (Disable)	0
enumdomain	The global default ENUM domain. By default the MSX uses <code>trial.e164.com</code> for all ENUM domain name resolutions. Endpoints can override this setting.	[Null]
err2isdn-abandoned	ISDN cause code sent on leg1 if call fails due to this internal error	16
err2isdn-busy	ISDN cause code sent on leg1 if call fails due to this internal error	17
err2isdn-dest-gone	ISDN cause code sent on leg1 if call fails due to this internal error	22
err2isdn-dest-rel-comp	ISDN cause code sent on leg1 if call fails due to this internal error	31
err2isdn-dest-timeout	ISDN cause code sent on leg1 if call fails due to this internal error	42
err2isdn-dest-unreach	ISDN cause code sent on leg1 if call fails due to this internal error	27
err2isdn-disconnect-ureach	ISDN cause code sent on leg1 if call fails due to this internal error	102
err2isdn-fce-error	ISDN cause code sent on leg1 if call fails due to this internal error	34
err2isdn-fce-error-setup	ISDN cause code sent on leg1 if call fails due to this internal error	34
err2isdn-fce-no-vports	ISDN cause code sent on leg1 if call fails due to this internal error	34
err2isdn-general-error	ISDN cause code sent on leg1 if call fails due to this internal error	16

Attribute Name	Description	Default
err2isdn-gw-resource-unavailable	ISDN cause code sent on leg1 if call fails due to this internal error	47
err2isdn-h245-error	ISDN cause code sent on leg1 if call fails due to this internal error	127
err2isdn-h323-internal	ISDN cause code sent on leg1 if call fails due to this internal error	41
err2isdn-h323-maxcalls	ISDN cause code sent on leg1 if call fails due to this internal error	34
err2isdn-h323-proto	ISDN cause code sent on leg1 if call fails due to this internal error	41
err2isdn-hairpin	ISDN cause code sent on leg1 if call fails due to this internal error	91
err2isdn-incomp-addr	ISDN cause code sent on leg1 if call fails due to this internal error	28
err2isdn-invalid-phone	ISDN cause code sent on leg1 if call fails due to this internal error	28
err2isdn-local-disconnect	ISDN cause code sent on leg1 if call fails due to this internal error	16
err2isdn-max-call-duration	ISDN cause code sent on leg1 if call fails due to this internal error	0
err2isdn-msw-invalid-epid	ISDN cause code sent on leg1 if call fails due to this internal error	31
err2isdn-msw-notreg	ISDN cause code sent on leg1 if call fails due to this internal error	31
err2isdn-msw-routecallgk	ISDN cause code sent on leg1 if call fails due to this internal error	31
err2isdn-network-error	ISDN cause code sent on leg1 if call fails due to this internal error	34
err2isdn-no-bandwidth	ISDN cause code sent on leg1 if call fails due to this internal error	34
err2isdn-no-call-handle	ISDN cause code sent on leg1 if call fails due to this internal error	41

Attribute Name	Description	Default
err2isdn-no-error	ISDN cause code sent on leg1 if call fails due to this internal error	-1
err2isdn-no-nat-t-license	ISDN cause code sent on leg1 if call fails due to this internal error	34
err2isdn-no-ports	ISDN cause code sent on leg1 if call fails due to this internal error	34
err2isdn-no-route	ISDN cause code sent on leg1 if call fails due to this internal error	34
err2isdn-no-route-at-dest	ISDN cause code sent on leg1 if call fails due to this internal error	3
err2isdn-no-vports	ISDN cause code sent on leg1 if call fails due to this internal error	34
err2isdn-reject-route	ISDN cause code sent on leg1 if call fails due to this internal error	34
err2isdn-resource-unavailable	ISDN cause code sent on leg1 if call fails due to this internal error	47
err2isdn-shutdown	ISDN cause code sent on leg1 if call fails due to this internal error	31
err2isdn-sip-auth-req	ISDN cause code sent on leg1 if call fails due to this internal error	31
err2isdn-sip-forbidden	ISDN cause code sent on leg1 if call fails due to this internal error	31
err2isdn-sip-int-error	ISDN cause code sent on leg1 if call fails due to this internal error	31
err2isdn-sip-not-impl	ISDN cause code sent on leg1 if call fails due to this internal error	31
err2isdn-sip-proxy-auth-req	ISDN cause code sent on leg1 if call fails due to this internal error	31
err2isdn-sip-redirect	ISDN cause code sent on leg1 if call fails due to this internal error	31
err2isdn-sip-service-unavailable	ISDN cause code sent on leg1 if call fails due to this internal error	31



Attribute Name	Description	Default
err2isdn-switchover	ISDN cause code sent on leg1 if call fails due to this internal error	31
err2isdn-temporarily-unavailable	ISDN cause code sent on leg1 if call fails due to this internal error	17
err2isdn-undefined	ISDN cause code sent on leg1 if call fails due to this internal error	31
err2isdn-user-blocked	ISDN cause code sent on leg1 if call fails due to this internal error	21
err2isdn-user-blocked-at-dest	ISDN cause code sent on leg1 if call fails due to this internal error	21
err2sip-abandoned	SIP response code sent on leg1 if call fails due to this internal error	500
err2sip-busy	SIP response code sent on leg1 if call fails due to this internal error	486
err2sip-dest-gone	SIP response code sent on leg1 if call fails due to this internal error	500
err2sip-dest-rel-comp	SIP response code sent on leg1 if call fails due to this internal error	500
err2sip-dest-timeout	SIP response code sent on leg1 if call fails due to this internal error	503
err2sip-dest-unreach	SIP response code sent on leg1 if call fails due to this internal error	480
err2sip-disconnect-ureach	SIP response code sent on leg1 if call fails due to this internal error	480
err2sip-fce-error	SIP response code sent on leg1 if call fails due to this internal error	500
err2sip-fce-error-setup	SIP response code sent on leg1 if call fails due to this internal error	500
err2sip-fce-no-vports	SIP response code sent on leg1 if call fails due to this internal error	500
err2sip-general-error	SIP response code sent on leg1 if call fails due to this internal error	503

Attribute Name	Description	Default
err2sip-gw-resource-unavailable	SIP response code sent on leg1 if call fails due to this internal error	503
err2sip-h245-error	SIP response code sent on leg1 if call fails due to this internal error	503
err2sip-h323-internal	SIP response code sent on leg1 if call fails due to this internal error	503
err2sip-h323-maxcalls	SIP response code sent on leg1 if call fails due to this internal error	503
err2sip-h323-proto	SIP response code sent on leg1 if call fails due to this internal error	503
err2sip-hairpin	SIP response code sent on leg1 if call fails due to this internal error	500
err2sip-incomp-addr	SIP response code sent on leg1 if call fails due to this internal error	484
err2sip-invalid-phone	SIP response code sent on leg1 if call fails due to this internal error	484
err2sip-local-disconnect	SIP response code sent on leg1 if call fails due to this internal error	503
err2sip-max-call-duration	SIP response code sent on leg1 if call fails due to this internal error	500
err2sip-msw-invalid-epid	SIP response code sent on leg1 if call fails due to this internal error	503
err2sip-msw-notreg	SIP response code sent on leg1 if call fails due to this internal error	503
err2sip-msw-routecallgk	SIP response code sent on leg1 if call fails due to this internal error	503
err2sip-network-error	SIP response code sent on leg1 if call fails due to this internal error	503
err2sip-no-bandwidth	SIP response code sent on leg1 if call fails due to this internal error	503
err2sip-no-call-handle	SIP response code sent on leg1 if call fails due to this internal error	481

Attribute Name	Description	Default
err2sip-no-error	SIP response code sent on leg1 if call fails due to this internal error	500
err2sip-no-nat-t-license	SIP response code sent on leg1 if call fails due to this internal error	500
err2sip-no-ports	SIP response code sent on leg1 if call fails due to this internal error	503
err2sip-no-route	SIP response code sent on leg1 if call fails due to this internal error	404
err2sip-no-route-at-dest	SIP response code sent on leg1 if call fails due to this internal error	404
err2sip-no-vports	SIP response code sent on leg1 if call fails due to this internal error	503
err2sip-reject-route	SIP response code sent on leg1 if call fails due to this internal error	404
err2sip-resource-unavailable	SIP response code sent on leg1 if call fails due to this internal error	480
err2sip-shutdown	SIP response code sent on leg1 if call fails due to this internal error	500
err2sip-sip-auth-req	SIP response code sent on leg1 if call fails due to this internal error	401
err2sip-sip-forbidden	SIP response code sent on leg1 if call fails due to this internal error	403
err2sip-sip-int-error	SIP response code sent on leg1 if call fails due to this internal error	500
err2sip-sip-not-impl	SIP response code sent on leg1 if call fails due to this internal error	501
err2sip-sip-proxy-auth-req	SIP response code sent on leg1 if call fails due to this internal error	407
err2sip-sip-redirect	SIP response code sent on leg1 if call fails due to this internal error	500
err2sip-sip-service-unavailable	SIP response code sent on leg1 if call fails due to this internal error	503

Attribute Name	Description	Default
err2sip-switchover	SIP response code sent on leg1 if call fails due to this internal error	500
err2sip-temporarily-unavailable	SIP response code sent on leg1 if call fails due to this internal error	480
err2sip-undefined	SIP response code sent on leg1 if call fails due to this internal error	503
err2sip-user-blocked	SIP response code sent on leg1 if call fails due to this internal error	403
err2sip-user-blocked-at-dest	SIP response code sent on leg1 if call fails due to this internal error	403
faststart	Send out fast-start Setup to terminating H.323 endpoint. Valid values: 1 (Enable)   0 (Disable)	1
first-authpass	Authentication password for first server. Used only if billing type is ciscoprepaid for sip calls.	[Null]
first-authuser	Authentication user name for first server. Used only if billing type is ciscoprepaid for sip calls.	[Null]
first-radsecret	The authentication/encryption key shared between the MSX and the second RADIUS server. Making a change to this requires an MSX restart.	[Null]
first-radserver	The IP/FQDN of the first RADIUS server to which the client will send accounting records. Making a change to this requires an MSX restart.	[Null]
forward-src-addr	Enabling this attribute causes the host name received in the "From" header of an incoming INVITE to be passed to the destination unchanged. Disabling this attribute causes the host address to be replaced with the RSA of the destination endpoint. Valid values: 1 (Enable)   0 (Disable).	0

Attribute Name	Description	Default
fsinconnect	Convey whether the fast start can be carried in Connect. Valid values: 1 (Enable)   0 (Disable).	0
fwname	Firewall type name (none, MS). Making a change to this requires an MSX restart.	none
g711alaw64k-duration	Use this buffer duration in milliseconds to forward in the H.323 codec capability list for g711alaw for SIP-to-H.323 calls.	20
g711ulaw64k-duration	Use this buffer duration in milliseconds to forward in the H.323 codec capability list for g711ulaw for SIP-to-H.323 calls.	20
g7231-frames	Number of g711alaw frames to forward in the H.323 codec capability list for SIP-to-H.323 calls.	1
g729-frames	Number of g711ulaw frames to forward in the H.323 codec capability list for SIP-to-H.323 calls.	2
getanifromacf	Extract the new ANI from the non-standard data in an ACF message received from a master gatekeeper. Valid values: 1 (Enable)   0 (Disable)	0
gkid	Identifier for MSX when it is acting as a gatekeeper. Gateways attempting to register with the MSX may need this ID configured on them for the registration to succeed.	[Null]
h245-tunneling	Use H.245 tunneling. Valid values: 1 (Enable)   0 (Disable)	1
h323cps	Maximum number of incoming ARQs per second the MSX will accept. The MSX will send ARJs when this limit is reached and log the rejected calls.	200

Attribute Name	Description	Default
h323-infotranscap	Information transport capability sent by the MSX in the Q.931 portion of outgoing Setup message. Options are [pass   speech   unrestricted   restricted   audio   unrestrictedtones   video]	pass
h323-instance	Specifies how many instances of the H.323 stack will be running on the MSX. Default is 1. Changing this parameter is not recommended except under the direction of NexTone Support. Making a change to this requires an MSX restart.	1
h323-maxbuffersize	Maximum buffer size used for decoding H.225/H.245 messages. Making a change to this requires an MSX restart.	4096
h323-maxcalls	Number of concurrent H.323 call legs. Making a change to this requires an MSX restart.	200
h323-maxcalls-pad-fixed	Stop accepting/making new H.323 calls if we hit a limit where we have only 200 of the configured H.323 max calls left. This parameter is pre-configured by NexTone and should not be changed.	200
h323-maxcalls-pad-variable	Stop accepting/making new H.323 calls if we hit a limit where we have only 20% of the configured H.323 max calls left. This parameter is pre-configured by NexTone and should not be changed.	80
h323-maxcalls-sgk	Number of concurrent call legs involved in calls going through a master gatekeeper in multi-instance mode. Making a change to this requires an MSX restart.	100
h323-maxrasbuffsize	Maximum buffer size used for decoding RAS messages. Making a change to this requires an MSX restart.	2048
h323-removetcs2833	Remove RFC2833 capability from an outgoing TCS. Valid values: 1 (Enable)   0 (Disable)	0

Attribute Name	Description	Default
h323-removetcst38	Remove T.38 fax capability from an outgoing TCS. Valid values: 1 (Enable)   0 (Disable)	0
hairpin	Allow hairpin calls i.e. calls to and from the same gateway. Valid values: 1 (Enable)   0 (Disable)	1
hdebug-CM	Entry for H323 stack module CM. Valid values: 1 (Enable)   0 (Disable)	0
hdebug-CMAPI	Entry for H323 stack module CMAPI. Valid values: 1 (Enable)   0 (Disable)	0
hdebug-CMAPICB	Entry for H323 stack module CMAPICB. Valid values: 1 (Enable)   0 (Disable)	0
hdebug-CMERR	Entry for H323 stack module CMERR. Valid values: 1 (Enable)   0 (Disable)	0
hdebug-level	H323 stack log level needs to be set to appropriate level for logging information to identify and troubleshoot problems. Multiple Levels of logging are available: warn(1), error(2), debug(3), trace(4)	0
hdebug-LI	Entry for H323 stack module <b>LI</b> . Valid values: 1 (Enable)   0 (Disable)	0
hdebug-LIINFO	Entry for H323 stack module <b>LIINFO</b> . Valid values: 1 (Enable)   0 (Disable)	0
hdebug-PERERR	Entry for H323 stack module <b>PERERR</b> . Valid values: 1 (Enable)   0 (Disable)	0
hdebug-TPKTCHAN	Entry for H323 stack module <b>TPKTCHAN</b> . Valid values: 1 (Enable)   0 (Disable)	0
hdebug-UDPCHAN	Entry for H323 stack module <b>UDPCHAN</b> . Valid values: 1 (Enable)   0 (Disable)	0
hidesrcrsa	Valid values: 1 (Enable)   0 (Disable)	0
inochserver-addr	IP address of the Inoch server used to do LNP (local number portability) dips.	[Null]

Attribute Name	Description	Default
inochserver-port	UDP port on which to contact the Inoch server.	0
inochserver-timeout	Number of milliseconds to wait before concluding that the Inoch server is unavailable.	50
interface-monitor-list	Interfaces that should be monitored for system health, in a redundant system.  A space-separated list of interface names in quotes. For example, "eth0" "eth1". Changing this requires an MSX restart.	[Null]
interimcdrtimer	Time interval in seconds at which interim CDRs will be written. If this is 0 there will be no interim CDRs.	0
internalifs	Comma-separated string of interface names on which there will be no firewalling. There can be no space between two interface names. For example: eth0,eth1,eth2. Requires MSX restart.	all
iserverlc.xml	The iserverlc.xml license file	(n/a)
jitter-buffer	Jitter buffer value in milliseconds.	80
leading-plus-sign-in-uri	Strip leading plus sign from phone number in SIP URI. Valid values: 1 (retain a plus sign)   0 (strip out plus sign)	1
local-proceeding	Send local call proceeding to originating H.323 endpoint after receiving a Setup from it. This is useful where the call is likely to be hunted multiple times and there is a possibility of the originating endpoint timing out and abandoning the call. Valid values: 1 (Enable)   0 (Disable).	0
local-reinvite-no-sdp	Respond to a SIP reinvite without SDP with a local 200Ok. The default behavior is to relay the reinvite through to the destination endpoint. Valid values: 1 (Enable)   0 (Disable).	0



Attribute Name	Description	Default
mapisdncc	Enable ISDN cause code mapping on a global basis. Valid values: 1 (Enable)   0 (Disable).	0
maplrjreason	Map LRJ reason code undefined to request denied. Valid values: 1 (Enable)   0 (Disable).	0
matchsipauthuser	Match username in RADIUS ProxyAuthorization or Authorization header with user name in call's From header. Valid values: 1 (Enable)   0 (Disable).	0
maxcallduration	If a call is active for longer than this time in seconds, it is torn down.	0
maxhuntallowdur	Limit defining how long a call is allowed to hunt. When this limit (in seconds) is reached, hunting is stopped and the call is rejected.	0
maxhunts	Limits the number of hunts allowed for a call originated by an endpoint. Limit is 50. A value of 1 disables hunting.	1
mdevices.xml	The mdevices.xml file.	(n/a)
memwrapper	Use the NexTone memory wrapper. Making a change to this requires a restart. Valid values: 1 (Enable)   0 (Disable). Making a change to this requires an MSX restart.	0
mgmt-ip	Management IP address. Making a change to this requires an MSX restart.	[Null]
mswname	Local host name.	(n/a)
nafAfModelNumber	The model number of the iServer as it is configured on the DF server.	[Null]
nafAfSerialNumber	The serial number of the iServer as it is configured on the DF server.	[Null]

Attribute Name	Description	Default
nafIsAuthenticationRequired	Flag indicating whether authentication is required after MSX connects to the DF server. <b>This flag should be set to 1 for SS8 DF servers.</b>	0
nafisSSLRequired	Flag indicating if call data and provisioning channels must be secured. <b>This should be set to 0 for SS8 DF servers.</b>	0
nafLocalAfTransportPort	The local transport port number for the iServer, for example, 20110. The iServer is the access function (AF) server.	0
nafLocalMediaSenderUdpPort	The iServer uses the port specified to send media to the DF server. It can be any value other than 20000, for example, 20140.	0
nafRealmName	The name of the realm to use to communicate with the DF server. The RSA of this realm and the port specified below as the Local AF transport port are used to send provisioning responses, call signaling and call content messages from the access function (AF) server, which is the iServer, to the DF server.  Using a separate realm for AF-DF communication is recommended.	[Null]
nafRemoteDfHostName	The IP address or hostname of the DF server from which provisioning data (warrants and collectors) will be received. If a name is specified it should be resolvable through /etc/hosts or DNS.	[Null]
nafRemoteDfTransportPort	The IP port number corresponding to the remote DF host, above.	0
nafx509CertFile	Self X.509 certificate file, qualified with the full path to the file. The file should be stored so that it is not accessible to users without LI privileges if <code>nafisSSLRequired</code> is set to a non-zero value.	[Null]

Attribute Name	Description	Default
nafx509CertParaphrase	X.509 certificate paraphrase. This is required only if nafisSSLRequired is set to a non-zero value.	[Null]
nonceduration	The nonce expiry duration for SIP digest authentication.	0
nonstdisdncc	Decode the ISDN CC present in the non-standard data portion of an LRJ/ARJ received from a Cisco peer/master gatekeeper. Valid values: 1 (Enable)   0 (Disable)	0
obp	Enable OBP mode. For this to work, SIP server type must be set to proxystateful. Valid values: 1 (Enable)   0 (Disable)	0
obpxfactor	This parameter in seconds is used by the MSX when it is operating in OBP mode to throttle keep-alive SIP Register messages from dynamic endpoints to the remote registrar.	900
pass-display-name-unchanged	Supports the "pass the display name unchanged" function. 1=enable, 0=disable.	0
pass-subjdata	Pass application specific subject data in SIP and H.323 messages. Valid values: 1 (Enable)   0 (Disable)	0
peer-callid	Print the peer leg call ID in field 37 of the CDR. Valid values: 1 (Enable)   0 (Disable)	0
peer-iserver	IP address of the control interface of the peer server. Making a change to this requires an MSX restart.	[Null]
podport	Specifies the port number to which the RADIUS server must send its POD packets. Making a change to this requires an MSX restart.	1700
podserverkey	The shared secret text string between the MSX and the RADIUS server. Making a change to this requires an MSX restart.	[Null]

Attribute Name	Description	Default
podsupport	Enable or disable packet-of-disconnect (POD) support. Valid values: 1 (Enable)   0 (Disable). Making a change to this requires an MSX restart.	0
podusername	The userid the RADIUS server sends to the MSX when sending POD packets. Making a change to this requires an MSX restart.	[Null]
policing	Enable RTP bandwidth policing to limit bandwidth used by calls. Valid values: 1 (Enable)   0 (Disable)	0
purgingwindow	This attribute represents the purging window for deleting records from the msw_journal table.	1
q931resptimeout	Response timeout in seconds for outgoing Q.931 requests. Making a change to this requires an MSX restart.	5
radacct	Enable RADIUS accounting on the MSX. Making a change to this requires an MSX restart. Valid values: 1 (Enable)   0 (Disable)	0
raddeadtime	The duration in seconds that a RADIUS server is considered unavailable when it times out based on radtimeout and radtries before attempting to reach it again. Making a change to this requires an MSX restart.	0
raddir	The path to the directory into which RADIUS log files are placed. Default is /usr/local/nextone/bin. Making a change to this requires an MSX restart.	/usr/local/nextone/bin

Attribute Name	Description	Default
radretries	The number of times a RADIUS request is resent to a RADIUS server before concluding the server is not available. This is usually invoked when a server is not responding or responding too slowly as defined by radtimeout. Making a change to this requires an MSX restart.	4
radtimeout	The interval in seconds that the MSX waits for a RADIUS server to reply. Applies to all RADIUS servers. Making a change to this requires an MSX restart.	5
rasmaxretries	Maximum number of times a RAS request is transmitted. Making a change to this requires an MSX restart.	1
rasresptimeout	Response timeout in seconds for outgoing RAS requests. Making a change to this requires an MSX restart.	5
record-route	Insert Record-Route header field into the dialog so that future requests in that dialog are routed through the MSX. Valid values: 1 (Enable), 0 (Disable)	0
route-call	Attempt to route calls to a destination endpoint. Valid values: 1 (Enable)   0 (Disable)	1
route-h245	Route H.245 signaling. Valid values: 1 (Enable)   0 (Disable)	1
rrqtimer	Time interval in seconds at which the MSX sends lightweight keep-alive RRQs to a master gatekeeper. Making a change to this requires an MSX restart.	30
sdebug-level	Sip stack log level needs to be set to appropriate level for logging information to identify and troubleshoot problems. Multiple Levels of logging are available: warn(1), error(2), debug(3), trace(4)	0

Attribute Name	Description	Default
second-authpass	Authentication password for second server. Used only if billing type is ciscoprepaid for sip calls.	[Null]
second-authuser	Authentication user name for second server. Used only if billing type is ciscoprepaid for sip calls.	[Null]
second-radsecret	The authentication/encryption key shared between the MSX and the second RADIUS server. Making a change to this requires an MSX restart.	[Null]
second-radserver	The IP/FQDN of the second RADIUS server to which the client will send accounting records in case the first RADIUS server fails to respond. Making a change to this requires an MSX restart.	[Null]
segowner	This is an MSX shared memory setting and is pre-configured as the owner of shared memory segments (gis/cli). Making a change to this requires an MSX restart.	
segs	This is an MSX shared memory maximum segments setting and is pre configured. Owner of shared memory segments (gis/cli). Making a change to this requires an MSX restart.	
sendirr	Send an unsolicited IRR to the master gatekeeper when an H.323 call leg is connected and continue to respond to the IRQs of gatekeeper with solicited IRRs as long as the call leg remains connected. Valid values: 1 (Enable)   0 (Disable).	0
server-type	Enable redundancy/peering and stateful call migration. Valid values (active   disabled). Making a change to this requires an MSX restart.	disabled

Attribute Name	Description	Default
sipauth	Enable or disable support for SIP authentication. Valid values: none   local   radius	none
sipauthpassword	Authentication password to use if sip-authentication type is local.	[Null]
sipconvertholdtoby	Convert SIP hold reinvoke to SIP Bye (used for Skype). Valid values: 1 (Enable)   0 (Disable).	0
sipdelay	Introduces a delay sometimes required with Polycom phones when using shared call appearances while running in OBP mirror proxy mode. Recommended value if using Polycom phones to do shared-call appearance is 200 milliseconds.	0
siphold3264	For IWF calls, support SIP hold and resume as per RFC3264 i.e. by sending a media attribute with direction SendOnly in the hold Invite and direction RecvOnly in the hold 200Ok. Valid values: 1 (Enable)   0 (Disable).	0
sipmaxforwards	Maximum number of times a SIP request can be forwarded.	70
sipminse	Minimum session timer expiry value accepted by the MSX.	600
sipoptionscodeclist	Codec list for SIP OPTIONS response.	[Null]
sipport	UDP port used to receive SIP messages.	5060
sipqlen	Limits the number of pending SIP call setups. May be useful in combating denial-of-service (DOS) attacks. This value should not be altered except on direction of NexTone Support.	1000
sipservertype	The SIP server type. If obp is to be enabled, SIP server type must be proxystateful. Valid values: redirect   proxy   proxystateful.	proxystateful

Attribute Name	Description	Default
sipsess	SIP session timer expiry value in seconds, set by the MSX.	3600
siptimer-C	SIP Timer C value in seconds.	180
siptimer-I	This timer in seconds defines how long a server transaction can remain in the Confirmed state before it times out and is terminated. On the MSX it applies only when the MSX is running in OBP mode and the proxy challenges Invites.	5
siptimer-shorthunt	The time interval in seconds for the MSX acting as a B2BUA to wait before retrying an outgoing SIP request with a redundant endpoint.	10
siptimer-T1	SIP TimerT1 value in microseconds.	500000
siptimer-T2	SIP TimerT2 value in microseconds.	4000000
siptrans-invitec	Maximum number of SIP Invite retransmissions.	7
sipusecontactintohdr	Use Contact in To header in SIP messages. Valid values: 1 (Enable)   0 (Disable).	0
sipusediversion	Use SIP diversion header information to route a call. Valid values: 1 (Enable)   0 (Disable).	0
stophuntrxlcfc	Stop hunt to the next peer GK if the first one responds to an LRQ with an LCF. Valid values: 1 (Enable)   0 (Disable).	0
synctimer	Timer value to synchronize the dynamic information of EndPoints, IEdgeGroups, and stats in shared memory with the database.	5
tagging	Enable 802.1p/q priority and IP type-of-service TOS-tagging. Valid values: 1 (Enable)   0 (Disable).	0



Attribute Name	Description	Default
tcpunreachcc	If the MSX times out after initiating an outgoing H.323 call it will treat the event as if it had received the ISDN cause code specified in this setting from the destination endpoint.	0
threads-bridge	Number of Bridge threads. This parameter is pre-configured by NexTone and should not be changed. Making a change to this requires an MSX restart.	10
threads-h323	Number of H.323 threads. This parameter is pre-configured by NexTone and should not be changed. Making a change to this requires an MSX restart.	1
threads-iwf	Number of IWF threads. This parameter is pre-configured by NexTone and should not be changed. Making a change to this requires an MSX restart.	1
threads-stack	Stack threads value. This parameter is pre-configured by NexTone and should not be changed. Making a change to this requires an MSX restart.	256
threads-tsm	Number of SIP TSM threads. This parameter is pre-configured by NexTone and should not be changed. Making a change to this requires an MSX restart.	2
threads-ua	Number of SIP UA threads. This parameter is pre-configured by NexTone and should not be changed. Making a change to this requires an MSX restart.	2
use3261branch	Use the RFC3261 format to generate a SIP branch tag. Valid values: 1 (Enable)   0 (Disable).	1
usecodemap	Codemap file to use for cause code mapping and hunting. e.g. if you specify usecodemap 002, codemap_002.dat will be used.	001

Attribute Name	Description	Default
use-ip-ani-auth	A billing category parameter. Valid values: 1 (Enable)   0 (Disable).	0
usexconnid	Enable support for X-Connection-Id SIP header. Valid values: 1 (Enable)   0 (Disable).	0

### Setting SIP parameters

Using `nxconfig.pl` to set SIP parameters is detailed in *Configuring global SIP parameters* on page 212.

### Configuring optional H.323 parameters

Using `nxconfig.pl` to configure optional H.323 parameters (H.323 fast start, force H.245, H.245 routing, etc.) is detailed in *Using nxconfig.pl to configure H.323 optional parameters* on page 275.

### Setting the global default ENUM domain

The MSX uses the `trial.e164.com` domain by default for all ENUM domain name resolutions, for all endpoints not configured with their own `enumdomain`. If you are using a different domain for default ENUM name resolution, add it to the MSX either by provisioning an endpoint of type ENUM Server in RSM Console, or by using this CLI procedure:

1. Log on to the MSX.
2. Start the `nxconfig.pl` utility by entering:

```
nxconfig.pl -e enumdomain -v enum domain name
```

 where `enum domain name` is the name of an ENUM name resolution domain you want the iServer to use.

### Enabling the Firewall Control Entity (FCE) feature

To enable the FCE feature on the MSX and set the firewall name using `nxconfig.pl`, follow these steps:

1. Log on to the MSX.
2. Start the `nxconfig.pl` utility by entering:

```
nxconfig.pl -E
```

3. Press <Return> repeatedly until the following prompt appears:

```
Firewall:
-----
fwname <none|MS>:
    a) => none
    b) => MS
    q) => quit
```

The current setting appears within the square brackets. Type a different value if you wish to change it, or press *q* to accept the default, then press <Enter>.

4. Press <Enter> at all subsequent prompts to accept the default values until the `nxconfig.pl` script completes.

### Setting QoS parameters

Quality of Service (QoS) support is available on MSXs with NSF-NP installed and named as the current firewall<sup>1</sup>. For a description of QoS, see *QoS (Quality of Service) metrics CDR fields* on page 423.

To set parameters for QoS using the `nxconfig.pl` utility:

1. Log on to the MSX.
2. Start the `nxconfig.pl` utility by entering:
 

```
nxconfig.pl -E
```
3. Press <Return> repeatedly until the following prompt appears:

```
QoS:
-----
tagging? <0|[1]>:
```

The current setting for tagging is shown in square brackets. Press <Enter> to keep the current setting, or change it by typing 0 to disable QoS tagging, or 1 to enable it, then press <Enter>.

4. Next is the setting for RTP bandwidth policing:

```
policing? <0|[1]>:
```

Press <Enter> to keep the current setting, or change it by typing 0 to disable policing, or 1 for enable it, then press <Enter>.

5. Finally you can set the value for the Jitter Buffer:

---

1. Note that tagging and RTP bandwidth policing only work when individually licensed. See *QoS licensing* on page 104.

```
jitter-buffer [2-255]? <80>:
```

Press <Enter> to keep the current setting, or change it by typing a jitter buffer value from 2–255, then press <Enter>. Note that this value is divided by 2 for each call leg.

6. Press <Enter> at all subsequent prompts to accept the default values until the `nxconfig.pl` script completes.

### **MSX peering and database replication parameters**

The MSX can be set up in a “high availability” configuration to ensure a greater overall level of service availability through redundancy of servers and data. The `nxconfig.pl` utility can configure two options connected with redundant systems: peering and database replication. For more information on this topic, see “MSX Redundancy”, on page 153.

To view and/or alter the settings affecting this feature, use the following procedure:

1. Log on to the MSX.
2. Start the `nxconfig.pl` utility. Enter:
3. Press <Return> repeatedly until the following prompt appears:

```
nxconfig.pl -E

iServer Peering Configuration:
-----

server-type: <disabled|active>:
```

The current setting appears within the square brackets. Type the other value if you wish to change it. Press <Enter>. If the active value is entered, `nxconfig.pl` presents the following additional prompts (if not, skip to step 4):

- 3.1 Enter the hostname for control-interface: <hostname or IP>:  
Redundant systems use an interface for communicating strictly between the servers comprising the redundant pair. This should be a separate interface from those used for call signaling and media.  
Enter the interface’s name here, using one of the interfaces listed in the prompt, and press <Enter>.
- 3.2 Enter the hostname for peer-iserver:  
Enter the IP address of the control interface of the MSX with which this MSX is redundantly paired.
- 3.3 Enter the hostname for interface-monitor-list?:

This prompt, requests the names of the interfaces on the MSX to monitor. A failure on one of these interfaces causes a failover to occur. Enter the names of one or more interfaces, each within quotation marks and separated by spaces, then press <Enter>.

**Caution:** *The control interface specified in step 3.1 must not have monitoring turned on. Failure to observe this precaution will result in service interruption in the event of an active processor failure.*

**Note:** *Altering the interface-monitor-list takes effect after an MSX restart. Restarting the MSX drops H.323 calls and all pending call setups.*

4. Press <Enter> at all subsequent prompts to accept the default values until the nxconfig.pl script completes. When prompted to commit changes, enter y.

### **MSX management interface configuration**

This final *servercfg* attribute is where you specify the IP address of the interface on this machine that RSM Console will connect to as its management interface. At the prompt:

1. Log on to the MSX.
2. Start the nxconfig.pl utility. Enter:
3. Press <Enter> repeatedly until the following prompt appears:

```
System
-----
.
.
.
mgmt-ip [nnn.nnn.nnn.nnn]:
```

The current setting is shown in square brackets. Press <Enter> to keep the current setting, or change it by typing the correct IP address of the MSX's management interface.

4. Press <Enter> at all subsequent prompts to accept the default values until the nxconfig.pl script completes.

## Global configuration using RSM Console

This section describes all configuration items found under RSM Console's iServer > Configure... menu. These items are generally referred to as the "global" MSX configuration parameters.

Table 4 lists the global configuration items and provides a description of each.

**Table 4. Global Configuration Items**

Item	Description
<b>SIP</b>	
Server Name	The MSX's SIP server name. Commonly displayed as the Owner Username in SDP messages. Default is "NexTone-MSX".
Authorization and Authorization Password	If "Authorization" is set to "radius" and a password is configured in the "Authorization Password" field, SIP user agents not registered using this password will have their invites rejected.
Server Type	The MSX can be configured as any of the three server types in this pull-down list. Most people will want their MSX to be configured as "Stateful Proxy" to take full advantage of the MSX's functionalities.
Record Route	<i>Note: This option is not supported in this release. Please leave it unchecked.</i>  This check box controls the SIP "Record-Route" function. If checked, the MSX inserts the Record-Route header field into the dialog, forcing future requests in that dialog to be routed through the MSX.
OBP	Enables OBP mode. For this to work, the SIP Server Type value must be "proxystateful".
Dynamic Endpoints	Allows dynamic endpoint registration in OBP mode.
NAT Traversal	Enables NAT traversal in OBP mode.
Maximum Forwards	Configures the maximum number of forwards allowed for a call. Prevents a call from entering into an infinite loop.
SIP Timer C	The number of seconds set for Timer C. See <i>SIP Timer C</i> on page 91 for details.
SIP Timer Short Hunt	The time interval in seconds for the MSX acting as a B2BUA to wait before retrying an outgoing SIP request with a redundant endpoint.
SIP Port	The port number on which the MSX will receive SIP INVITEs from endpoints.

Item	Description
SIP Qlen	Limits the number of pending SIP call setups. Adjusting this value may be useful in combating denial-of-service (DOS) attacks. This value should not be altered except on direction of NexTone Support.
<b>H.323</b>	
Gatekeeper ID	This is the MSX's ID when acting as a gatekeeper. Gateways attempting to register with the MSX may need this ID configured on them for the registration to succeed.
RRQ Timer	The time interval at which the MSX sends lightweight RRQs to a Master Gatekeeper when registering to it, to indicate that the MSX is still alive.
Q931 Response Timeout	Response timeout (in seconds) for outgoing Q.931 requests.
RAS Response Timeout	Response timeout (in seconds) for outgoing RAS requests.
RAS Maximum Try	Maximum number of times a RAS request is transmitted. <i>Note: This parameter should not be set to a value greater than 5 unless directed by NexTone Support.</i>
Calls Per Second	Used to throttle call volume. This is an estimated maximum number of calls per second the MSX will handle. The MSX starts rejecting calls when this rate is reached. All dropped calls are recorded in the CDR.
Instances	Specifies the number of instances of the MSX running on the MSX platform. You must stop and re-start the MSX for this change to take effect!
Master GK Max Calls	Only valid when MSX Instances > 1. The total number of estimated calls that the MSX will send/receive from all provisioned Master Gatekeepers. Calculated on a per-leg basis. (e.g. For 500 calls between two MGK's, this setting would be 1000. For 500 calls between one MGK and some other device, this setting would be 500.)
Max Calls	The maximum number of H.323 call legs (not end-to-end calls) the system can process. We recommend this value to be a little more than twice the number of "vports" your license allows. (e.g. If you have purchased 1000 vports, it can be set to 2500.)
Info Trans Cap	Information Transport Capability. See <i>Information transfer capability</i> on page 274 for more information.
Route H.245	ENABLED (checked) is the only supported option

Item	Description
Call Routing	Recommended setting is ENABLED (checked). If this option is DISABLED, the MSX will NOT route any H.323 calls. It would simply be used as a stateless GK.
Local PROCEEDING	Recommended setting is ENABLED (checked). Enabled for call hunting purposes when codec info is present in Call Proceeding from failed attempts at destination. See <i>Configurable local proceeding</i> on page 280 for more information.
Allow Dest. ARQ	Recommended setting is DISABLED, to maintain best security. For details on this setting, see <i>Registration and setup flow</i> on page 261.
H.245 Tunnel	Determines whether H.245 tunneling is used. H.245 tunneling tunnels the H.245 session through the H.225 connection by embedding H.245 messages in H.225 messages.
Allow Auth. ARQ	Authenticate all originating H.323 endpoints with the master gatekeeper.
<b>FCE (FIREWALL CONTROL ENTITY)</b> (See “Firewall Control Entity (FCE)”, on page 334 for information about this page)	
Firewall: Enable Media Services Firewall	Select this to enable firewalling on media services devices.
Internal Interfaces	Internal interfaces to exclude from firewalling
Add Device Add Pool Add Vnet	See the procedures in <i>Firewall Control Entity (FCE)</i> on page 334 for information about these controls
configuration panes (3)	RSM Console lists the currently-defined MSX devices, routing pools, resource pools, and media Vnets. See the procedures in <i>Firewall Control Entity (FCE)</i> on page 334 for information about these panes.
<b>BILLING</b> (See the chapter, <i>Billing and CDR Processing</i> on page 393 for CDR details)	
Billing Type	Currently the MSX only supports “Post-paid” and “Cisco-Prepaid”.
Prepaid Authentication User	RADIUS authentication user name. Used only if billing type is ciscoprepaid for sip calls. See “”, on page 364 for information about authentication. This field is active if the “prepaid” or “ciscoprepaid” billing type is selected.
Prepaid Authentication Password	Password for user ID specified in <b>First User</b> . This field is active if the “prepaid” or “ciscoprepaid” billing type is selected.



Item	Description
Prepaid Authorization User	User name for RADIUS authorization. Used only if billing type is ciscoprepaid for sip calls. See “”, on page 364 for information about authentication. This field is active if the “prepaid” or “ciscoprepaid” billing type is selected.  This field is active if the “prepaid” or “ciscoprepaid” billing type is selected.
Prepaid Authorization Password	Password for user ID specified in <b>Authorization User</b> . This field is active if the “prepaid” or “ciscoprepaid” billing type is selected.
CDR Type	Fixed: one file for all CDRs  Daily: one file per day, closed at midnight  Seq: fixed file size, each file is 1 MB  Time: the CDR file closes at the time interval specified in <i>CDR Timer</i> , below.
CDR Timer	The time interval in minutes for <i>CDR type</i> of “Timer”
CDR Interim	The time interval, in minutes, after which the first interim CDR will be generated for a call. After that, additional interim CDRs will be generated at intervals of CDR Interim/2 minutes.
Leg 1 Start	Checking this option configures the MSX to record “Leg 1 Start” info in the .ctt and .ctr files. (See the chapter, <i>MSX Configuration</i> for detailed information on call legs and logging options.)
Leg 1 End	The standard CDR files (.cdr and .cdt) always record the “Leg 1 End” info, since the MSX calculates call duration from this time. This option cannot be un-checked.
Leg 2 Start	Checking this option configures the MSX to record “Leg 2 Start” info in the .ctt and .ctr files.
Leg 2 End	Checking this option configures the MSX to record “Leg 2 End” info in the .ctt and .ctr files.
Hunt	Checking this option configures the MSX to record “Call Hunt” info in the .ctt and .ctr files.
Radius: Primary Server	The address (IP or FQDN) of the primary server to which RADIUS accounting records are sent.
Radius: Primary Secret	The shared authentication and encryption key for all communications between the RADIUS client and the RADIUS server.
Radius: Secondary Server	The address (IP or FQDN) of an alternate server to which RADIUS accounting records are sent when the primary is unavailable.

Item	Description
Radius: Secondary Secret	The shared authentication and encryption key for the secondary RADIUS server.
Radius: Timeout	The interval (in seconds) a the MSX waits for a server host to reply. Applies to all RADIUS servers.
Radius: Retry	The number of times a RADIUS request is re-sent to a server before concluding the server is not available.
Radius: Dead Time	A server is considered unavailable this long when it times out (based on <i>Radius: Timeout</i> and <i>Radius: Retry</i> ) before attempting to reach it again.
Send RADIUS Accounting Messages	Checking this option enables use of MSX's RADIUS communication capabilities for CDR spooling. See <i>Accounting</i> on page 364 for more information.
Use Overloaded Session ID Format	Checking this option selects XA3+ format for RADIUS communication.
Enable POD	Enable or disable packet-of-disconnect (POD) support. POD is a feature where a RADIUS server sends a network packet to the MSX to force a particular in-progress call to disconnect immediately.
POD port	The port number to which the RADIUS server must send its POD packets
POD username	The user ID the RADIUS server sends to the MSX when sending POD packets.
POD password	The password for the POD user ID.
<b>REDUNDANCY</b>	
Note that this tab only allows viewing selected redundancy-related parameters. See Chapter 9, "MSX Redundancy," on page 153 for more info on Redundancy.	
<i>Network tab</i>	
Status	The status of network redundancy. If this is set to <i>disabled</i> , the other fields shown on this panel are disabled as well.
Control Interface	The MSX interface used to communicate with the peer (redundant) MSX. This is usually the Private (that is, not Public-side) network interface, unless a second private interface is dedicated specifically for this feature.
Peer iServer Control IP	The redundant (Peer) MSX's Control Interface IP address.
Interface Monitor List	The list of interfaces that the MSX monitors for failure.

Item	Description
<b>SYSTEM</b>	
<i>Calls tab</i>	
Enable Call Hunting	This check box enables the MSX's call hunting ability system-wide. It affects all endpoints with a call hunt setting of "Inherit System Default".
Maximum Call Hunts	Only relevant when call hunting is enabled. Specifies the maximum number of call hunting attempts for endpoints with a call hunt setting of "Inherit System Default". The maximum number of hunts is "50". A value of "1" means "No Hunt".
Maximum Call Hunts Duration	Only relevant when call hunting is enabled. This optional parameter limits how long a call can be hunting for an endpoint to complete through. When the limit (in seconds) is reached, the call is rejected.
Allow All Sources	This check box enables any source to originate calls using this MSX. (See <i>Allow All Sources</i> on page 61 for details.)
Forward Source Address	When this option is enabled, the MSX forwards the source signaling address to the destination in all IWF (SIP-H.323 Interworking Function) calls.
Hairpin Calls	Checking this box enables hairpin calls, which are calls that have their source and destination in the same gateway.
Remove RFC2833	Selecting this option removes the assertion of RFC2833 capability (DTMF tones encoded into RTP packets) in an "H.245 fast start", thereby allowing for H.245 negotiation of the RFC 2833 codec capability.
Remove TCST38	Selecting this option removes the assertion of TCST38 fax capability in an "H.245 fast start", thereby allowing for H.245 negotiation of the TCST38 fax capability.
Maximum Call Duration	The limit on how long a call is permitted to last before being forcibly torn down. Used most often to thwart fraud, the number should be large, if used.
<i>Codec tab</i>	
g711 U-law Duration	The MSX uses this buffer duration (packet time, in milliseconds) to forward in the H.323 codec capability list for SIP-to-H.323 IWF calls.
g711 A-law Duration	The MSX uses this buffer duration (packet time, in milliseconds) to forward in the H.323 codec capability list for SIP-to-H.323 IWF calls.

Item	Description
g729 Frames	The MSX uses this number of frames (packet time) to forward in the H.323 codec capability list for SIP-to-H.323 IWF calls.
g723.1 Frames	The MSX uses this number of frames (packet time) to forward in the H.323 codec capability list for SIP-to-H.323 IWF calls.
Default	Use this pull-down to specify the codec for this MSX when performing H.245 “slow start” negotiating. During fast start, the value supplied in the setup is passed through.
<i>Other tab</i>	
ENUM Domain	By default, the MSX uses trial.e164.com for all ENUM domain name resolutions. You can specify a different domain to use here. See <i>Setting the global default ENUM domain</i> on page 50. Individual endpoints can override this setting.
Cache Timeout	<p>The life of a dynamic registration, before which an endpoint must re-register. The actual working value is the minimum of:</p> <ul style="list-style-type: none"> <li>• This value</li> <li>• The value the endpoint proposes at registration, and</li> <li>• The value a third-party server assigns, if one is involved.</li> </ul> <p>See <i>Endpoint registration</i> on page 224 for details.</p>
OBP Registration Scaling Factor	The number of seconds used by the MSX when it is operating in OBP mode to throttle keep-alive SIP Register messages from dynamic endpoints to the remote registrar.
Server Name	The hostname of this MSX
Management IP	The IP address of the MSX network interface used for remote logins.
Use code map	The three digit sequence number of the code map to use. See “Cause Code Operations,” on page 524 for a discussion of code maps.
Map ISDN cause code	Enables ISDN cause code mapping on a global basis.
Map LRJ reason code	Enable to map the “undefined” LRJ reason code to “request denied.”
Send IRR	Send an unsolicited IRR to the master gatekeeper when an H.323 call leg is connected and continue to respond to the IRQs of gatekeeper with solicited IRRs as long as the call leg remains connected.
Inoch Server	The IP address, port and timeout values for the Inoch server that endpoints will use for LNP dipping.

Item	Description
ADVANCED	
Number of Segments	These settings are related to the MSX shared memory and other internal settings. These are pre-configured when the MSX is first set up and there is no need for them to be changed, unless instructed by a NexTone technical support engineer. Improper settings may cause the server to malfunction.
Thread Stack Size	
SIP TSM Threads Pool	
LOGGING	
Logs tab	
Log Path	The directory where the MSX debug log files are placed.
H.323 Log Path	The directory where the MSX H.323 log files are placed.
Custom tab	
SIP Stack Debug Level	Sets the SIP stack debug level. This setting should be set to <b>off</b> except when instructed by a NexTone technical support engineer, as it can adversely affect the MSX's performance.
H.323 Stack Debug Level	Sets the H.323 stack debug level. This setting should be set to <b>off</b> except when instructed by a NexTone technical support engineer, as it can adversely affect the MSX's performance.
debug checkboxes	The checkboxes in the “Custom” tab are used for advanced troubleshooting. Please do not enable these unless instructed by a NexTone technical support engineer, as it can adversely affect the MSX's performance.

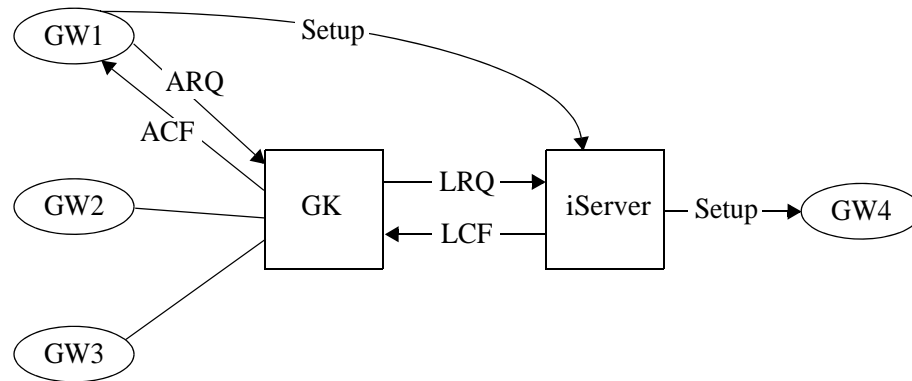
### ***Allow All Sources***

Normally, *Allow All Sources* is disabled. In this configuration, when a call setup arrives at the MSX, the MSX checks its own database to see if the source is a configured endpoint. If the endpoint is *not* provisioned, the MSX rejects the call.

Enabling *Allow All Sources* permits any gateway to make calls *through* the MSX, even if it's *not provisioned on* the MSX. This option is useful when a call originates from a device registered to a stateless gatekeeper provisioned as an endpoint on the MSX (such as GW1 is in Figure 3). The call would normally be rejected, since the source gateway sends its Setup message directly to the MSX (on which it is not provisioned). Enabling "Allow All Sources" tells the MSX to accept call setups from any gateway (even one *not* registered on the MSX).

Figure 3 illustrates the “Allow All Sources” signaling flow.

**Figure 3. Allow All Sources**



If you do enable “Allow All Sources,” you still have some options to filter out unwanted calls. You can:

- Use a call’s destination prefix.
- Configure a large number of endpoint IP addresses, if necessary. The GenEP application, provided by NexTone, generates endpoint lists in text format for importation into the database.

Another approach is to disable “Allow All Sources,” and provision the “Subnet IP and Subnet Mask” under the Advanced tab behind the gatekeeper.

#### **Setting “Allow All Source Numbers” using nxconfig**

You can enable this option by using the nxconfig utility. The steps are:

1. Log on to the MSX.
2. Change to the directory where the nxconfig utility is stored, and start it:

```
nxconfig.pl -E
```

3. Press <Enter> repeatedly until the following prompt appears:

```
Calls
-----
```

```
allow-src-all <[0]|1>
```

The current setting is shown in square brackets. Press <Enter> to keep the current setting, or change it by entering a 0 to disallow calls from all sources, or 1 to allow calls from all sources.

4. Press <Enter> at all subsequent prompts to accept the default values until the `nxconfig.pl` script completes.

### **Call hunting with multiple directory gatekeepers**

Every call that hits the MSX goes to the directory gatekeeper with an LRQ, and the gatekeeper sends back an LCF with up to three destinations, consisting of a gateway IP address and a number to dial.

A company may have MSXs co-located with DGKs. Additionally, it may have backup DGKs at other locations.

Multiple gatekeepers may refer to a single route server database. Therefore, once the three destinations given by the first directory gatekeeper do not result in a completed call, hunting on backup gatekeepers will produce the same result.

Consequently, you can set the `stophuntrxlcfc` attribute to “enabled” to refuse further hunting. To enable this attribute:

1. Log on to the MSX.
2. Enter:

```
nxconfig.pl -e stophuntrxlcfc -v 1
```

### **Local Number Portability (LNP) support**

The MSX supports an endpoint performing an LNP server “dip” to get LNP data. The feature is configured at two levels: The MSX, and the endpoint. Presently, only an “Inoch Server” is supported. For information on configuring the endpoint side of the setup, see *Enabling LNP for an endpoint* on page 92.

The customer's Inoch server is specified and configured using the `nxconfig.pl` utility, as follows:

1. Log on to the MSX.
2. Set the Inoch server IP address by entering:
 

```
nxconfig.pl -e inochserver-addr -v inoch ip addr
```

 where inoch ip addr is the IP address of the Inoch server.
3. Set the Inoch server port. This is the port to contact on the Inoch server when dipping it. Enter:

```
nxconfig.pl -e inochserver-port -v inoch port
```

where inoch port is the port number.

4. Set the Inoch server timeout period. This is the number of milliseconds to wait before concluding that the server is unavailable. Enter:

```
nxconfig.pl -e inochserver-timeout -v timeout
```

where timeout is the timeout period.

### **RSM Console procedure**

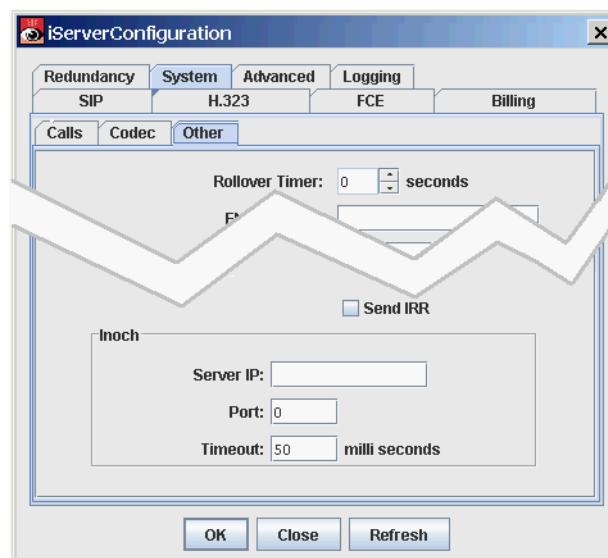
1. Open a web browser on a Windows or X-Windows workstation and in the Address field, enter:

```
https://rsm ip/rsm
```

where rsm ip is the address of the RSM workstation managing the MSX, if the MSX is being managed by an RSM workstation; otherwise, specify the IP address of the MSX's management interface.

2. Start RSM Console, and in its map window, right-click on the MSX you wish to configure. Select Configure... The **iServerConfiguration** window appears (Figure 4).

**Figure 4. Entering Inoch Server Parameters**



3. Scroll down if necessary to reveal the Inoch frame. Supply the parameters in the appropriate text boxes.
4. Click OK to save your changes, then Close, or just click Close to discard any changes and exit the window.



## Procedure: add an endpoint

This section details the steps in configuring an endpoint. It begins with a list of endpoint device types and a brief comparison of master and peering gatekeeper types. It then adds a table of data items, with a description of each, and questions to ask in order to obtain the needed item. Finally, a flowchart titled *Adding an Endpoint* is provided as the last page of this chapter (page 95). The chart lists device types and the configuration data required for each.

### Determining the device type

The first step in adding a new endpoint to an MSX is to determine which type of device you are adding. This information is based on the network layout and the specific installed devices.

The MSX software supports the following device types:

- Generic IP Device (H.323/SIP IP Phone)
- H.323 IP Phone
- SIP IP Phone
- H.323 Gateway (GW)
- H.323 Gatekeeper (GK), also called “Peer Gatekeeper”
- H.323 Master Gatekeeper (MGK, or Sgatekeeper)
- SIP Gateway/Proxy
- Softswitch (SIP or H.323)
- ENUM server
- MSX server
- Position server

The section *What to configure, and what to ask* on page 66 provides a list of configuration parameters, and a flowchart to follow, based on the endpoint device type. Table 5 in that section gives a definition for each parameter.

Note that all endpoint types require the parameters listed in the *General Configuration* box, to the left on the flowchart page. The boxes to the right list General Configuration as a starting point, followed by the additional information needed for the type of endpoint being configured. Also note that the *H.323 General Configuration data* is also required for all H.323 endpoints.

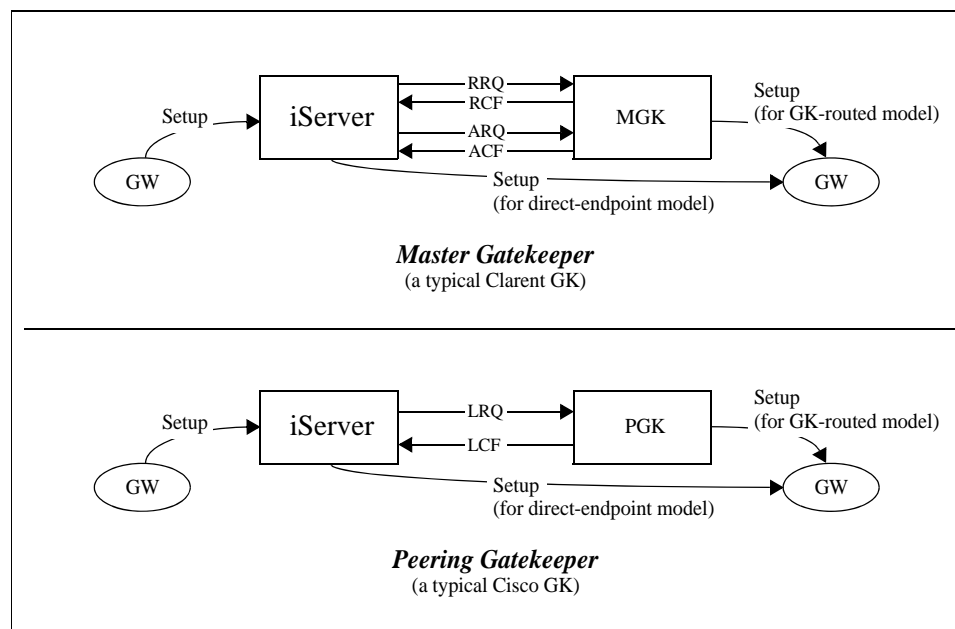
The third column in Table 5 gives additional information about where the data may be obtained, who is responsible for determining how to set that parameter, and some hints about what it might be set to and why.

The RSM Console screens have configuration fields corresponding to the fields' check boxes listed on the flowchart.

### **Master gatekeeper or peering gatekeeper**

Note that if the endpoint is a gatekeeper (GK), it can have one of two relationships with the MSX. It can be either a *Master Gatekeeper* (MGK), or *Peering Gatekeeper* (PGK). Figure 5 depicts these two relationships, showing the differences between them.

**Figure 5. MSX-to-Gatekeeper Relationships**



If a gatekeeper is configured as a Master GK in the MSX, the MSX registers to that MGK using an H.323 ID assigned from the MGK. It is also necessary to configure the MGK's GK ID on the MSX as the "Peer GK ID".

A gatekeeper should be configured as a Master GK in the MSX whenever the gatekeeper requires the MSX to register to it, or if the gatekeeper is unable to support gatekeeper peering (as with the Clarent gatekeeper). When set up this way, the MSX acts as a gateway to the MGK.

### **What to configure, and what to ask**

Once the endpoint type information is obtained, and the list of required parameters determined from the flow chart on page 95, use the information in Table 5

to obtain the actual data you will enter into the RSM Console configuration tabs. For example, all endpoints require a Registration ID, and the table tells you that it is a unique ID used by MSX, and that it is supplied by the MSX administrator.

**Table 5. Endpoint Configuration Data Items**

Item	Description	Question to ask / note	CLI keyword
<b>Phone TAB</b>			
Partition	The partition to which the endpoint belongs	Does this endpoint belong to a partition other than "admin," and if so, which one?	pid, must be an integer from 1 to 64. See RSM partitioning.
Device Type	The kind of endpoint being added/configured	What kind of device is this (IP phone, GW, GK, etc.)?	type
Registration ID	A unique ID used internally by the MSX to identify an endpoint.	MSX admin should assign this ID to the endpoint	regid
Port Number	A virtual port number for the endpoint. In RSM Console, port numbering always starts at "1" and the numbers are automatically generated during the port creation process. (Note that port numbering in the MSX database, and therefore in cli commands, begins at zero.)	Auto-numbered. Can add a port manually for multiple calling plans, alternative Master GK or multiple subnet access list configurations	uport
IP address	IP address of the endpoint. IP address is not required, when an endpoint is registering with its H.323 ID or E.164 address.	IP address of the endpoint	ip
Extension	This is the E.164 address of an endpoint. This is only available to "Generic IP Device" types (IP Phones).	For an IP Phone, the question is "what is the E.164 number (or simply phone number) of the IP Phone?"	phones
Calling Plan	This is the calling plan this endpoint uses. To have multiple calling plans, the endpoint should have multiple virtual ports configured.	MSX admin should define the calling plan and calling routes according to the customer's requirement.	cp

Item	Description	Question to ask / note	CLI keyword
Realm	The realm in which this endpoint resides. (See Chapter 11, “MSX Realms”). <sup>a</sup>	In what realm is this endpoint located?  Note that a realm called “any” should not be used for VoIP gateways, which do not register, and therefore must be defined as being in a specific realm.	realm
iEdge Group	The group limit to which this endpoint contributes and subscribes (if any). (See <i>U-port group limits</i> on page 80.)	Does this endpoint belong to an <i>igrp</i> , and if so, which one?	igrp
Enable Transcoding	Enables the use of media transcoding for the endpoint	Does this endpoint require transcoding?	transcode
Codec Profile	Specifies the Codec Profile the endpoint uses.	What transcoding capabilities does the endpoint require? Match this to the appropriate Codec Profile.	cdcpfl
<b>Advanced TAB</b>			
Zone	MSX uses the “Zone” to group endpoints. If a zone is configured, the endpoint can make calls only to other endpoints with the same zone or to endpoints within the “Null” (empty) zone. An endpoint with a Null zone can only talk to other endpoints with “Null” zone.	MSX admin should define the zone.	n/a
Vendor	To achieve interoperability with different vendor's products, the MSX needs to know some endpoint (product)'s vendor. These include Clarent and Sonus. Most vendor products can be set to “Generic”.	What is the make and model of the endpoint device?	vendor

Item	Description	Question to ask / note	CLI keyword
Subnet IP Address and Subnet Mask	<p>The Subnet IP and Subnet Mask define a range of IP addresses that are allowed to set up calls to the MSX. This is useful when a call is originated from a GW registered to a stateless GK, and the GK in turn, sends traffic to the MSX (as Peer GK or MGK). In this set up, the MSX has no knowledge of the originating GW, therefore must do one of the following:</p> <p>Turn on the global "Allow All Source" setting.</p> <p>Configure an IP range to accept the call.<sup>b</sup></p>	<p>Does your GK support "GK Routed Signaling"? If not, can you provide us all the IP addresses and/or IP ranges of your gateways registered to your GK?</p> <p>Note that the IP range (defined by the Subnet IP Address and Subnet Mask) acts as an <i>access list</i> to filter out unwanted calls. To configure multiple entries, additional virtual ports must be added to the endpoint to define the new ACL entries. Large lists of IP addresses/Subnets can be added using an application called GenEP. (Contact NexTone Support for information about GenEP.)</p>	subnetip and subnetmask
Outgoing Prefix	For SIP calls, this optional prefix precedes the outgoing <i>RequestURI</i> and <i>To</i> field contents, whether or not the incoming setup includes this character or string of characters.	<p>Should the endpoint prepend a string to all outgoing SIP setups?</p> <p>Field not available for Generic IP Device and User Account endpoints.</p>	ogp
Domain Match			n/a
<b>User Info TAB</b>			
User Information	Information about the customer who owns or operates this endpoint.		n/a

Item	Description	Question to ask / note	CLI keyword
Customer ID	The customer ID is displayed in the CDR as an ID and used to identify a CDR's originating/destination endpoints.	Is this endpoint associated with only one customer, and if so, what is its customer ID?	n/a
<b>Protocol TAB</b>			
Gateway/ Proxy	This tells the MSX that the endpoint is either a gateway or a proxy, such as an H.323 gateway/gatekeeper or SIP gateway/proxy. This check box is automatically checked when the endpoint type is set as H.323 GW/GK, SIP GW/Proxy, ENUM Server, Position Server, MSX, or Soft Switch.	Check this option, only when a "generic IP device" is a gateway or proxy.	gw
Priority	The MSX considers this priority setting fifth in line when it searches for a destination route. MSX zone is first, followed by the route's active time-of-day, route binding priority, and destination pattern longest match, then this priority setting.	MSX admin should define this based on the desired routing requirements (e.g. Least Cost Routing).	priority
LNP	This enables endpoint-level Local Number Portability support. The MSX supports an endpoint performing an LNP server "dip" to get LNP data. The feature is configured at two levels, the MSX, and the endpoint. Presently, an "Inoch Server" is supported		n/a
SIP	Check this box if the endpoint supports SIP	Does this device support SIP?	sip
H.323	Check this box if the endpoint supports H.323	Does this device support H.323	h323
URI (SIP/H.323)			uri
<b>Note: For more information on the following trunk group supporting parameters, see Trunk group parameter descriptions on page 504</b>			
Src. Trunk Group	Limit call setups to those having a sourceCircuitID matching this string.		tg

Item	Description	Question to ask / note	CLI keyword
Dest. Trunk Group	For destination port selection, and to insert destination trunk information the into the egress leg.		dtg
New Source Ingress Trunk Group	Overrides or supplies missing source trunk information from the incoming setup.		newsrctg
New Source Egress Trunk Group	Overrides or supplies missing incoming leg destination trunk group information.		newsrctdg
Send Dest. Trunk Group	Controls the insertion of destination trunk information into the egress leg's setup message.		setdesttg
Remove Src. Trunk Group	Removes sourceCircuitID from egress leg setup requests		removetg
<b>Calls TAB</b>			
<b>Limit section<sup>c</sup></b>			
Maximum Total Calls	An optional limit on the maximum number of concurrent calls this endpoint can have.	MSX admin should configure this.	xcalls
Maximum Ingress Calls	An optional limit on the number of calls that can be placed from that endpoint to the MSX. This value is independent of the "Maximum Egress Calls" but limited by the "Maximum Total Calls".	MSX admin should configure this.	xincalls
Maximum Egress Calls	An optional limit on the number of calls that can be placed to that endpoint by the MSX. This value is independent of the "Maximum Ingress Calls" but limited by the "Maximum Total Calls".	MSX admin should configure this.	xoutcalls
<b>Hunting section</b>			
Enable Call Hunting	Enables call hunting for this endpoint. Hunting is a source-specific configuration which specifies the number of destination routes to be considered for call termination.	MSX admin should configure this.	n/a

Item	Description	Question to ask / note	CLI keyword
Inherit System Default	If this option is checked, this endpoint uses the system default settings for Maximum Call Hunts. The system default is configured in RSM Console on the iServer → Configure → System tab.	MSX admin should configure this.	n/a
Maximum Call Hunts	Limits the number of hunts allowed for a call originated by this endpoint. Limit is 50.	MSX admin should configure this.	maxhunts
<b>Media section</b>			
	<p>About Media Routing</p> <p>When a call is media-routed by the MSX, not just the signaling goes through the MSX, but the actual media traffic (in the form of RTP packets) goes through the MSX as well. A separate interface for media traffic is generally used to improve performance.</p> <p>Benefits of media routing include:</p> <ul style="list-style-type: none"> <li>• Hides source and destination IP addresses from each other</li> <li>• Minimizes carrier Access Control List provisioning</li> <li>• Enables public-to-private and private-to-public signaling/voice</li> </ul>		
Never Route Media	<p>Checked: this endpoint <i>will not</i> route media.</p> <p>Unchecked: this endpoint <i>will</i> route media.</p> <p>See Table 6 for details on these settings</p>	MSX admin should configure this.	nmr
Route Media	<p><i>Checked:</i> This endpoint will route media to/from other endpoints, <i>except</i> with ones that are explicitly configured as “Never Route Media.”</p> <p><i>Unchecked:</i> This endpoint will route media with endpoints on which “Route Media” is checked.</p>	MSX admin should configure this.	mr



Item	Description	Question to ask / note	CLI keyword
Mid-call Media Change	During a T.38 fax call setup, some gateways (such as Cisco) tear down the voice channel and start a new channel with a new set of RTP and RTCP ports. This causes some source gateways (Clarent's for example) to drop the call, since the gateway is not expecting mid-call media port and media control port changes. The MSX handles this situation, if "Mid-call Media Change" is checked, by maintaining the original RTP port when talking to the originating GW and using the new RTP port when talking to the destination GW.	MSX admin should configure this.	n/a
<b>H.323 CONFIGURATION</b>			
	<p>About H.323 Configuration</p> <p>The H.323 Configure button on the protocol page is active only if an endpoint is configured as an H.323-type endpoint (such as H.323 Gatekeeper, Master Gatekeeper, etc.) or an endpoint that can be SIP or H.323 (such as an ENUM server or MSX). Clicking this button displays the H.323 Protocol Parameters page described in this section.</p>		
GRQ	Applies to master gatekeeper endpoints only. Specifies whether the MSX should send a GRQ to this MGK (see page 272).		grq
RAI	Applies to master gatekeeper endpoints only. Specifies whether the MSX should send an RAI to this MGK.		rai
Apply Egress Routes...	Applies to master gatekeeper endpoints only.		n/a
Tech Prefix	Applies to master gatekeeper endpoints only. Specifies the technical prefix for the MSX to use when communicating with this MGK.		techp

Item	Description	Question to ask / note	CLI keyword
H.323 ID	The identifier assigned to a gateway that's registering to the MSX. It is configured into both the MSX (by the MSX administrator), and the gateway (by the gateway administrator).	If the endpoint is a registering GW, the MSX admin should create an H.323 ID for this endpoint and ask the customer to configure it on their gateway as well.	h323id
	If the endpoint is a <i>Master GK</i> , this is the H.323 ID the Master Gatekeeper assigns to the MSX.	If the endpoint is a MGK, ask, "What is the H.323 ID assigned to this MSX for it to use when registering with your GK?"	
Gatekeeper ID	Applies only to gatekeeper and master gatekeeper endpoint types. This is the regid that the MSX will use to communicate with this GK or MGK.		n/a
RAS Port	The endpoint's UDP port number, used by RAS messaging (RRQ/RCF/RRJ, ARQ/ACF/ARJ, LRQ/LCF/LRJ...). Typically UDP port 1719.	What is the RAS port number your endpoint uses (if it's not the standard port)?	rasport
Q.931 Port	The endpoint's TCP port number, used by the Q.931 call setup messages (Setup, Call Proceeding, Progress/Alerting, Connect). Typically TCP port 1720.	What is the Q.931 port number your endpoint uses (if it's not the standard port)?	q931port
Calling Party Number Type	Calling Party Number type to be sent in the Q.931 call setup message. The default setting is "Pass" (from the incoming setup). If set to another value, that value is forced into the setup.	MSX admin should configure this. Best to use the default unless it's not working.	cgpntype
Called Party Number Type	Called Party Number type to be sent in the Q.931 call setup message. The default setting is "Pass" (from the incoming setup). If set to another value, that value is forced into the setup.	MSX admin should configure this. Best to use the default unless it's not working.	cdpntype
Layer 1 Protocol	This configures the "Bearer Capability". The Bearer Capability from the ingress leg can be relayed (via Pass-Through) or configured to the egress leg during call setup. This configuration is specific to destination PSTN switches. The default is G711ulaw.	MSX admin should configure this. Please use the default unless it's not working.	bcaplayer1

Item	Description	Question to ask / note	CLI keyword
Info Transfer Cap	Sets the information transfer capability for this endpoint. (See <i>Information transfer capability</i> on page 274.)	What kinds of information will this device handle?	infotranscap
H.235 Password	TBS		passwd
Q.931 Display	This is the "Caller ID" string, consisting of any combination of alphabetic or numeric characters. If this option is <i>unchecked</i> , the MSX does not send this info, because some international PSTN switches drop calls that have Q.931 Display info.	MSX admin should configure this. Please use the default unless it's not working.	
Force H.245	If this option is <i>checked</i> , the MSX sends a "FACILITY" message to start an H.245 session. This is done as a last resort to start H.245 with devices that <i>do not</i> respond to the H.245 info (the IP & TCP port number) sent in the "CONNECT". This guarantees that there is always a H.245 session.	MSX admin should configure this.	forceh245
H.245 Address in Connect	If this option is <i>unchecked</i> , the MSX removes H.245 address info from the "CONNECT" message sent to the source. When setting up calls from the MSX to certain gateways, there is a chance of getting into a race condition when setting up H.245. This is caused by an H.245 TCP connection delay at the source and only occurs when the H.245 address is sent from the MSX in both the CONNECT and FACILITY messages. To avoid such race conditions, configure the MSX to send H.245 address only in a separate FACILITY message and not in the CONNECT.	MSX admin should configure this when H.245 is failing. A symptom of this is very short call durations (1-2 secs) to a particular GW.	connh245addr
ISDN CC Mapping	Turns on or off "Cause Code Mapping" for this endpoint.	Does this endpoint need ISDN cause code mapping?	mapcc

Item	Description	Question to ask / note	CLI keyword
PI on FastStart	An Alerting or Progress message in response to a “fast start” doesn’t always include a Progress indicator (PI). When it doesn’t, some endpoints won’t open a media channel for far-end ring tones. Selecting this option tells the MSX to insert a PI into the fast-start response.	Does this endpoint need PI forced on a fast-start Alerting or Progress message?	pionfaststart
<b>Remove from TCS section</b>			
RFC2833	Setting this option to “enabled” removes the assertion of RFC2833 capability (DTMF tones encoded into RTP packets) in an “H.245 fast start”, thereby allowing for H.245 negotiation of the RFC 2833 codec capability. Setting this option to “default” causes the global system setting for this parameter to be used		deltcs2833
T.38	Setting this option to “enabled” removes the assertion of T.38 fax capability in an “H.245 fast start”, thereby allowing for H.245 negotiation of the T.38 fax capability. Setting this option to “default” causes the global system setting for this parameter to be used		deltcst38
<b>SIP CONFIGURATION</b>			
<b>SIP section</b>			
Contact	This field is populated automatically when a SIP endpoint registers to the MSX. It has the format: phone_no@MSX_IP:5060.  If the SIP UDP port needs to be changed from 5060 to a non-standard port, it should be manually configured in the “Contact” field using the following format: phone_no@MSW_IP:non_standard_port.	What is your SIP contact (or URI)?	contact
Is 2833 Capable	Tells whether this endpoint supports RFC 2833 (DTMF via RTP). Options are yes, no and unknown (default).	This parameter must be set to yes or no for each endpoint, if the endpoint is to operate reliably.	2833capable

Item	Description	Question to ask / note	CLI keyword
Privacy	The kind of SIP privacy a destination endpoint supports. Options are RFC 3325, Draft 01, and both.	What SIP Privacy standard(s) do you want this endpoint to support?	privacy
2833 Payload Type	The RFC 2833 <i>payload type</i> for this endpoint.		pt2833
Caller ID Block	The default setting to control whether caller ID information is forwarded. Check this box to block caller ID forwarding. (This setting can be overridden for a call; see Table 14 on page 184.)		cidblock
FQDN redundancy	Used for configuring BroadSoft redundancy support. See <i>Broadsoft redundancy support</i> on page 208 for details.		
SIP host untrusted	Select this option if this endpoint is to be considered untrusted in SIP Privacy operations.		trust
Invite NOSDP	TBS		invitenosdp
Enable Recipient Update	TBS		recipientupdate
SIP Domain	TBS		sipdomain
<b>NAT section</b> (See <i>Configuring an endpoint for SIP</i> on page 207 for more information about these fields.)			
Automatic NAT Detection	Selecting this option enables automatic IP address and port detection for NAT traversal. If this is selected, the next two fields are disabled.		natdetect
NAT IP	If automatic NAT detection is disabled, the public-side IP address of the public-to-private router goes here.		natip
NAT Port	If automatic NAT detection is disabled, the public-side port to use with the above IP address goes here.		natport

- a. Note that a realm called “any” is available for call-originating or terminating endpoints, but it should not be used for gateways, which do not register, and therefore must be defined as being in a specific realm.

- b. The IP range (defined by the Subnet IP Address and Subnet Mask) acts as an access list to filter out unwanted calls. To configure multiple entries, additional virtual ports must be added to the endpoint to define the new ACL entries.  
Large lists of IP addresses/Subnets can be added using an application called GenEP. (Call NexTone for more information on GenEP.)
- c. For details on call limits, see “MSX Configuration,” on page 79.

**Table 6. Never Route Media and Route Media Endpoint Settings, Truth Table**

This Endpoint		Other Endpoint		Is Media Routed?
Never	Route	Never	Route	
—	—	✓	—	N
—	✓	✓	—	N
✓	—	—	—	N
✓	—	—	✓	N
☐	☐	☐	☐	N
☐	✓	☐	—	Y
☐	—	☐	✓	Y
☐	✓	☐	✓	Y

Key: ✓ = enabled  
 ☐ = disabled  
 — = either enabled or disabled (i.e., irrelevant)

### ***Making an endpoint “sticky”***

The endpoint *sticky* property affects hunting to that endpoint. If a call hunts to an endpoint that has its *sticky* parameter set to enable, the call either completes on that endpoint, or the call is rejected. A similar behavior occurs with the *sticky* property of a call route (see *Sticky routes* on page 482).

To make an endpoint sticky, you must use CLI, as this property is not supported with RSM Console. Log onto the MSX as `root` and use the `cli iedge edit` command, by entering:

```
cli iedge edit regid uport sticky enable
```

## Setting session limits on calls, by endpoint

Through a process known as *call admission control*, or CAC, the MSX gives its administrator fine control over the number of calls that a given endpoint may support. The maximum call limit of an endpoint defines its capacity for routing calls in the network, enhancing its usefulness for overall system load balancing and routing purposes. There are two measures of capacity: calls (used for registered gateways), and bandwidth (used for *un*registered devices).

The MSX has the capability to set independent session limits for three parameters at the gateway level:

- **Maximum Total Calls** – specifies the overall number of calls the gateway will support, both ingress and egress.
- **Maximum Ingress Calls** – specifies the maximum calls that may be placed from that endpoint/gateway to the MSX.
- **Maximum Egress Calls** – specifies the maximum number of calls that may be placed to that endpoint/gateway by the MSX.

In addition to limiting calls at the gateway level, the MSX can limit total bandwidth allocated at the subnet level for SIP endpoints. (Bandwidth limiting is not supported on H.323 endpoints). These parameters are:

- **Maximum Total Bandwidth** – specifies the overall bandwidth the subnet will support, both ingress and egress.
- **Maximum Ingress Bandwidth** – specifies the maximum bandwidth that may be placed from that subnet to the MSX.
- **Maximum Egress Bandwidth** – specifies the maximum bandwidth that the MSX may place to that subnet.

For more information on setting up subnet-based traffic limiting, see *Subnet CAC* on page 83.

These parameters are set from within RSM Console, using the **Modify** <endpoint> panel's Calls tab, for each endpoint, as shown below in Figure 6.

**Figure 6. Setting Session Call Limits for an Endpoint**

The screenshot shows the 'Provision an Endpoint' dialog box with the 'Calls' tab selected. The 'Limit' section contains three rows of call limits, each with a text input field and two radio buttons ('None' and 'Unlimited'). The 'Hunting' section includes two checked checkboxes ('Enable Call Hunting' and 'Inherit System Default') and a 'Maximum Call Hunts' spinner. The 'Media' section has three checkboxes: 'Never Route Media' (checked), 'Route Media', and 'Mid-call Media Change'. The dialog has 'OK' and 'Cancel' buttons at the bottom.

Note that *ingress* and *egress* are with respect to the MSX and not the endpoint/gateway. See *Call legs* on page 471 for clarification.

### **U-port group limits**

A “group limit” is provided for assigning registered endpoint/uports to a multi-device group, known as an *iEdge Group*, or *igrp*, to which multiple uports subscribe and collectively contribute. As with the session limits described in the prior section, this limit also is broken down by *incoming*, *outgoing* and *total* concurrent calls. iEdge groups also allow limitations on the amount of calling *bandwidth* (not just calls). This system of collective limits is administered either from within RSM Console, or from the CLI.



### **Administering iEdge groups with RSM Console**

RSM Console provides an interface for managing iEdge groups. It also has a selection pull-down for each endpoint, where you specify the igrp to which that endpoint subscribes and contributes.

#### ***The iEdge Groups utility***

Using this interface, you can add, change or delete iEdge groups. You can also monitor the current operating status of existing groups.

**Figure 7. iEdge Groups Utility Window**

iEdge Groups											
Group	Name	Max Calls In	Max Call...	Max Call...	Max Ban...	Max Ba...	Max Ban...	Timeout	imr	emr	Max Call...
admin	Grp 1	Unlimit	Unlimit	20	None	None	None	10	Ignore	Ignore	Wed Ma...
admin	Grp 2	15	15	30	None	None	None	10	Don't C...	Don't ...	Wed Ma...

You add and delete iEdge groups from the window's Edit menu. To change an existing group's properties, right click on that group's name, and choose Modify, or double-click the entry.

*Note: The item labeled "Max Calls Out Time" is the last time one or more calls were turned away because the limit was reached. It is initialized to the creation date of the iEdge Group.*

#### ***Adding and changing iEdge groups***

You create and change iEdge group parameters with the **Add iEdge Group** and **Modify iEdge Group** windows. See *iEdge Group parameters* on page 89.

***iEdge group subscription from the Modify Endpoint window***

Once the group exists, you use the Modify [endpoint type] panel Phone tab to subscribe one or more endpoints to it. Figure 8 shows this screen.

**Figure 8. Subscribing an Endpoint to an iEdge Group**

The screenshot shows a 'Provision an Endpoint' window with the 'Phone' tab selected. The 'iEdge Group' dropdown menu is highlighted with a grey oval, showing 'Grp 1' as the selected option. Other fields in the form include: Partition: admin, Device Type: H.323 Gateway, Registration ID: ABC Telco, Port Number: 0, IP Address: 10.10.10.10, Extension: (empty), Phone Number: (empty), VPN Phone Number: (empty), Calling Plan: <none>, Realm: <none>, Enable transcoding: (unchecked), and Codec Profile: <none>. The 'OK' and 'Cancel' buttons are at the bottom.

Choose an existing iEdge group from the pull-down highlighted in the above figure.

**Administering iEdge groups with CLI commands**

The general procedure for implementing igrps using CLI is to:

1. Create an iEdge group using `cli igrp add group name`
2. Set its limits with `cli igrp edit igrp name limit`
3. Subscribe a uport to the iEdge group with:  
`cli iedge edit regid uport igrp igrp name`

Additional limits are provided for emergency calling. See *iEdge group limits* on page 511.

See Appendix B, “The Command Line Interface”, for complete command descriptions and other commands for managing iEdge groups.

## **Subnet CAC**

### **Introduction**

Subnet-based Call Admission Control (CAC) is the ability to control call admission and media routing policy based on the subnet a call setup comes in from. This feature is useful in Tier 1 environments where a carrier doesn't want to maintain a static list of individual endpoints that may register with its MSX.

### **Overview**

Without this feature, only statically-provisioned endpoints<sup>2</sup> have a set of policies assigned to them at provisioning time. These policies control various endpoint-specific parameters, such as whether media is routed, the number of simultaneous ingress, egress, and total calls, etc. Subnet CAC extends these policies to undefined<sup>2</sup> endpoints, basing such application of policy on the subnet from which the endpoint registers.

### **CAC Policy**

A special type of named iEdge entity, the *policy*, supports subnet-level CAC. Assigning such control through a named set of parameters relieves the transport provider of the burden of manually maintaining a static database of all their customers' endpoints. As with statically-provisioned endpoints, the aggregate number of calls (or bandwidth demand, for subnets on SIP endpoints) that the MSX will allow is controlled for ingress, egress, and total, for that subnet. An iEdge Group (igrp) object is bound to the subnet policy, to control call admission and media routing.

### **Procedure**

The steps to creating and implementing a new iEdge policy are:

1. Identify or create an iedge group to use for this policy object.
2. Set the iedge group's policy parameters, if not already set.
3. Create a new, named iedge policy object.

---

2. There are really three classes of MSX network element. 1) Static: those manually provisioned, including their IP address, 2) Dynamic (permanent): those provisioned without an IP address, with policy tied to registration ID, and 3) Dynamic (transient): those not in the MSX's database at all (therefore, “undefined”). This feature applies policy to this third class, those not in the MSX's database, when the MSX is operating in OBP or mirror proxy mode.

4. Designate the object as `type = policy`.
5. Designate the policy type as `policy-key = subnet`.
6. Specify the subnet (IP address and subnet mask) to which the policy will apply.
7. Designate the realm to which this policy applies.
8. Designate the iEdge group you identified or created in step 1 as the source of CAC policy parameters for this policy object.

### **cli steps**

The commands to accomplish the above procedure are as follows.

1. Create a new iEdge group (skip this step if it already exists):

```
cli iedge add igrp iedgegroupname
```

2. Set the iEdge group's policy parameters (if they don't already exist):

```
cli igrp edit iedgegroupname parametername parametervalue 3
```

Valid parametername values are:

- `maxcallsin` (ingress calls limit)
- `maxcallsout` (egress calls limit)
- `maxcallstotal` (total [ingress + egress] calls limit)
- `maxbw` (ingress bandwidth limit; parametervalue is in bps)
- `maxbwout` (egress bandwidth limit; parametervalue is in bps)
- `maxbwtotal` (total [ingress + egress] bandwidth; parametervalue is in bps)
- `emr` (between iEdge groups routing [valid settings: xxx, alwayson, alwaysoff, on]) <sup>4</sup>
- `imr` (within iEdge groups routing [valid settings: xxx, alwayson, alwaysoff, on])

Example:

```
cli igrp edit T1-igroup maxbw 1544000
```

will set an ingress bandwidth limit 1.544 megabits per second on the iEdge group named T1-igroup.

3. Note that with numeric settings, a value of 0 (zero) means *unlimited*. To set a value of *none* (i.e., no calls/bandwidth), enter -1 as the value (which appears as 4294967295 in listings obtained with the `cli igrp lkup` or `list` commands).
4. See the *Internal and external media routing settings* on page 185 for descriptions and use of these settings.

3. Create a new iedge policy object (the trailing zero indicates uport 0, which policy objects always require):

```
cli iedge add polycyname 0
```

4. Set the type to policy:

```
cli iedge edit polycyname 0 type policy
```

5. Set the key to subnet:

```
cli iedge edit polycyname 0 policy-key subnet
```

6. Set the IP address for this policy:

```
cli iedge edit polycyname 0 subnetip subnetIPAddress
```

7. Set the subnet mask for this policy:

```
cli iedge edit polycyname 0 subnetmask subnetmask
```

8. Set the realm for this policy:

```
cli iedge edit polycyname 0 realm realmname
```

9. Subscribe to the iEdge group for this policy:

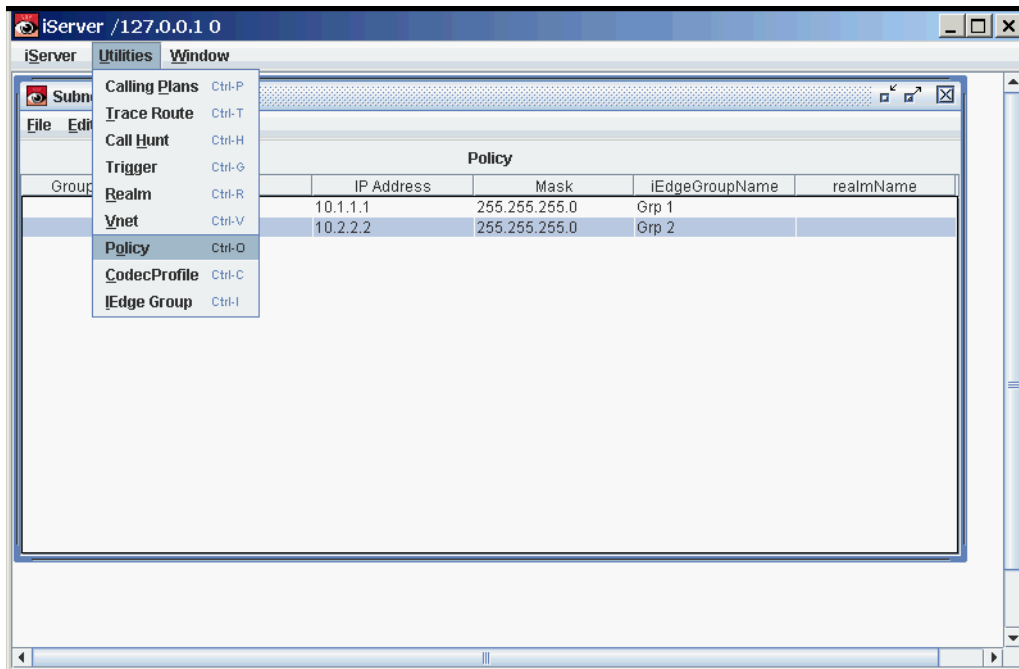
```
cli iedge edit polycyname 0 igrp iedgegroupname
```

### **RSM Console steps**

A new utility has been added to RSM Console's iServer window for creating and maintaining policy objects, to implement subnet-based CAC.

To access the utility, double-click the MSX's name in RSM Console's map window. When the **iServer** window displays, choose **Utilities** → **Policy**.

**Figure 9. Starting the Policy Utility**



The **Subnets** window appears (“subnet” is currently the only type of policy supported).

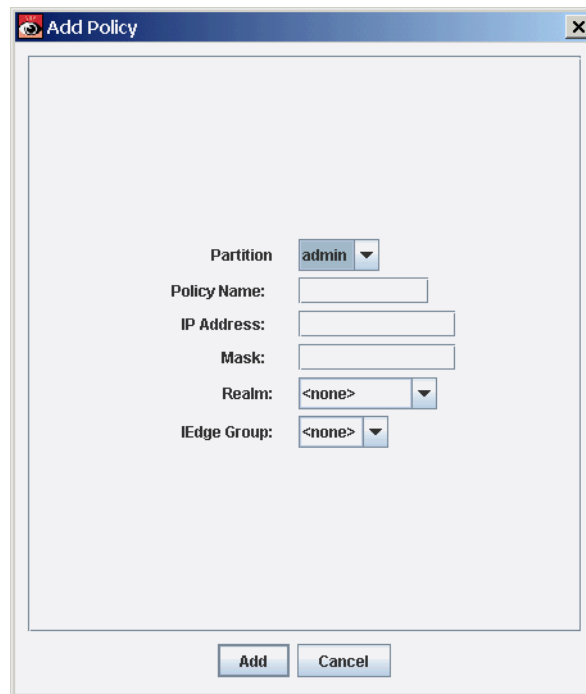
#### **Add a new Policy object**

Before creating a new policy object, ensure that the iEdge group and realm to which the new policy object will subscribe have already been created. If not, create them, then return here.

To create a new policy object, from the **Subnets** window:

1. Choose Edit → Add. The **Add Policy** window appears.

**Figure 10. Add Policy Window**



2. Select the partition to which the Policy applies.
3. Enter the Policy Name (max. 31 chars.), IP Address of the subnet to which this policy will apply, and the subnet mask for that subnet.
4. Select a predefined realm name to which endpoints on this subnetwork will belong, from the pull-down Realm list.
5. Select a predefined iEdge group name defining the policy to be applied, from the pull-down iEdge Group list.
6. Click Add to create the new policy, or Cancel to discard your work, and close the window.

**Change a Policy object**

To modify an existing policy object, from the **Subnets** window:

1. Locate and double-click the policy you wish to change. The **Modify Policy** window appears, showing the currently-defined parameters for that policy.

**Figure 11. Modify Policy Window**

The screenshot shows a window titled "Modify Policy" with a close button (X) in the top right corner. The window contains the following fields and controls:

- Partition:** A pull-down menu currently showing "admin".
- Policy Name:** A text field containing "p2".
- IP Address:** A text field containing "10.2.2.0".
- Mask:** A text field containing "255.255.255.0".
- Realm:** A pull-down menu currently showing "realm-internal".
- iEdge Group:** A pull-down menu currently showing "Grp 2".

At the bottom of the window, there are two buttons: "Modify" and "Cancel".

2. As required, change the partition to which the policy applies.
3. Note that you cannot change the Policy Name. You must delete and re-create a policy to change its name. Doing so will break existing references to the policy.
4. As required, change IP Address of the subnet to which this policy will apply, and the subnet mask for that subnet.
5. As required, change the predefined realm name to which endpoints on this subnetwork will belong, from the pull-down Realm list.
6. As required, change the predefined iEdge group name defining the policy to be applied, from the pull-down iEdge Group list.



- Click Modify to put your changes into effect, or Cancel to discard your changes, and close the window.

### *iEdge Group parameters*

To accommodate the policies supported, the **Add iEdge Group** and **Modify iEdge Group** windows have the form shown:

**Figure 12. iEdge Group Window**

Summary of the changes:

- **Max Call Legs In/Out/Total/None/Unlimited** - Set the limits on the number of call legs that a subnet can consume, along with block (None) and no limit (Unlimited) check-boxes.
- **Max Bandwidth In/Out/Total/None/Unlimited** - Set the limits on bandwidth, in bits-per-second, that a subnet can consume, along with block (None) and no limit (Unlimited) check-boxes. (Not enforced on H.323 endpoints.)
- **Media Routing** - Set policy on the kinds of realm-based media routing allowed. See *MSX Realms* on page 174 for details.
- **Max Calls Out Time** - This read-only field shows the last date and time that a Max Calls limit was reached/enforced. Formerly known as “Dnd Time.”

## Limiting bandwidth by call

The maximum bandwidth that a call may consume may be limited by enabling a feature known as *RTP Bandwidth Policing*. If policing is enabled, a call may not consume more bandwidth than it negotiated for, based on the codec used. If a call is found to be consuming more than the allowed amount, the extra packets are dropped by NSF-NP. RTP bandwidth policing is licensed feature (see *QoS licensing* on page 104). To enable or disable policing, run `nxconfig.pl` as described in *Setting QoS parameters* on page 51.

## Setting call duration limits

MSX provides two SIP-related *timers* to set limits on how long a call or call attempt may last before it is torn down: *maxcallduration*, and *siptimer-C*.

Note that the maximum call duration limit (*maxcallduration*) supersedes all other timers, including Timer C, if *maxcallduration* is set to a lower value than the others. *maxcallduration* is primarily to prevent certain types of errors or fraud from consuming resources without limit. Timer C is a way to handle certain errors that may occur while setting up SIP calls.

### **Maximum call duration**

This timer applies to both SIP and H.323 calls.

The maximum call duration timer places an upper limit on how long a call may remain in the system (either setting up or connected) before it is torn down as either fraudulent or abandoned.

This timer, *maxcallduration*, sets an absolute limit, in seconds, for any call, in any state. This timer ends any call after the number of seconds specified, and for this reason, is typically set to a large number, such as the number of seconds in 24 hours (86,400), since even a legitimate, in-progress call will end when this timer triggers.

To set this timer:

1. Log on to the MSX.
2. Set the *maxcallduration* value using the `nxconfig.pl` utility. Enter:

```
nxconfig.pl -e maxcallduration -v max_dur val
```

where max\_dur val is the value to assign as the maximum call duration value.

### **SIP Timer C**

SIP Timer C applies to SIP calls only. This timer begins at the start of the call attempt, with the MSX's reception of the first INVITE message on the ingress leg. From that point, the MSX must receive a message numerically-greater than a 100 (for example, TRYING, which doesn't reset the timer), before the timer times out, or else the call is torn down. A message greater than 100 indicates the call is ringing (180), has been set up (200), or cannot be set up (e.g., 503). A 180 (RINGING) does reset the timer to its value defined in *servercfg*.

If the call isn't established before the timer runs out, the call is torn down without further effort to establish it (even if it's ringing). Once the call *is* established, SIP timer C is ignored. This timer covers the scenario where a call setup/invite is attempted, but the call never completes, enabling the MSX to clear the uport for reuse.

To set this timer:

1. Log on to the MSX as root.
2. Set the *siptimer-C* value using the *nxconfig.pl* utility. Enter:

```
nxconfig.pl -e siptimer-C -v value
```

where value is the number of seconds to assign as the maximum call duration value.

### **Disabling media routing on specific endpoints**

Whether a call is media routed is based on the endpoint's stored configuration in the MSX. The functionality of the MSX is to media route all calls that have media routing enabled on at least one endpoint of the call. However, for peering purposes, a service provider may choose to not media route calls being terminated on provider owned gateways, even though the call may have originated on a partner owned gateway, which was set up to media route. By setting the flag "Never Route Media" on the provider-owned endpoint configuration, the provider can media route calls only between partners and not those being terminated on their network.

Note that this option exists only for gateways (SIP or H.323). This feature is available in the endpoint configuration in RSM Console, as shown in Figure 13.

**Figure 13. Configuring On-net Gateways to Never Route Media**

Modify H.323 Gateway: ABC Telco/1

Phone Advanced User Info Protocol **Calls**

**Limit**

Maximum Total Calls: 100 ☐ None ☐ Unlimited

Maximum Ingress Calls: 60 ☐ None ☐ Unlimited

Maximum Egress Calls: 60 ☐ None ☐ Unlimited

**Hunting**

☒ Enable Call Hunting

☒ Inherit System Default

Maximum Call Hunts: 1

**Media**

☒ Never Route Media

☐ Route Media

☐ Mid-call Media Change

OK Cancel

## Enabling LNP for an endpoint

In addition to configuring the MSX to support access to an Inoch server for local number portability (LNP) dips, each endpoint that will perform LNP dips must be configured to do so. Either the command-line interface (cli) or RSM Console can be used, as follows.

### CLI procedure

To enable LNP dipping with cli, enter:

```
cli iedge edit regid uport inoch enable
```

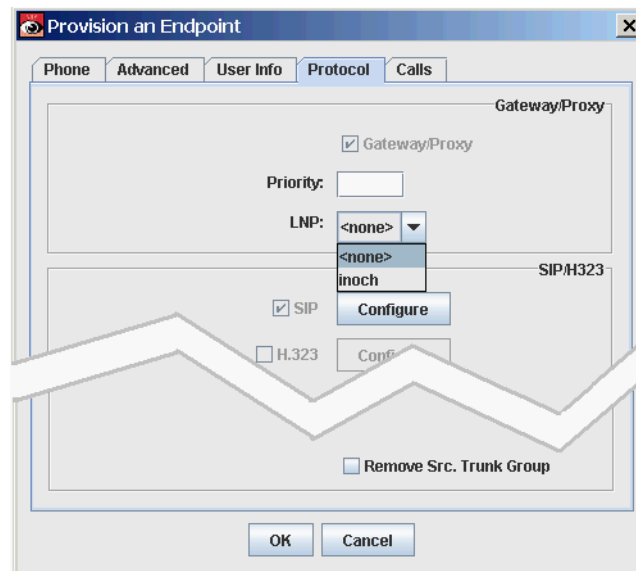
where regid and uport are the endpoint's registration ID and u-port to be enabled, respectively.

### **RSM Console procedure**

To enable this feature from RSM Console:

1. Start RSM Console and double-click the MSX on which the endpoint is configured.
2. Once the **Database** window is open, in the lower frame, locate the endpoint to configure. Click on the endpoint, then click Modify.
3. In the **Modify SIP Gateway** window that appears, click the Protocol tab. (Figure 14.)

**Figure 14. Configuring an Endpoint for LNP**



4. As shown in the figure, in the Gateway/Proxy frame, click on the LNP pull-down list, and select inoch.
5. Click OK to save your changes, or click Close to discard any changes and exit the window.

### **Setting the ENUM domain for an endpoint**

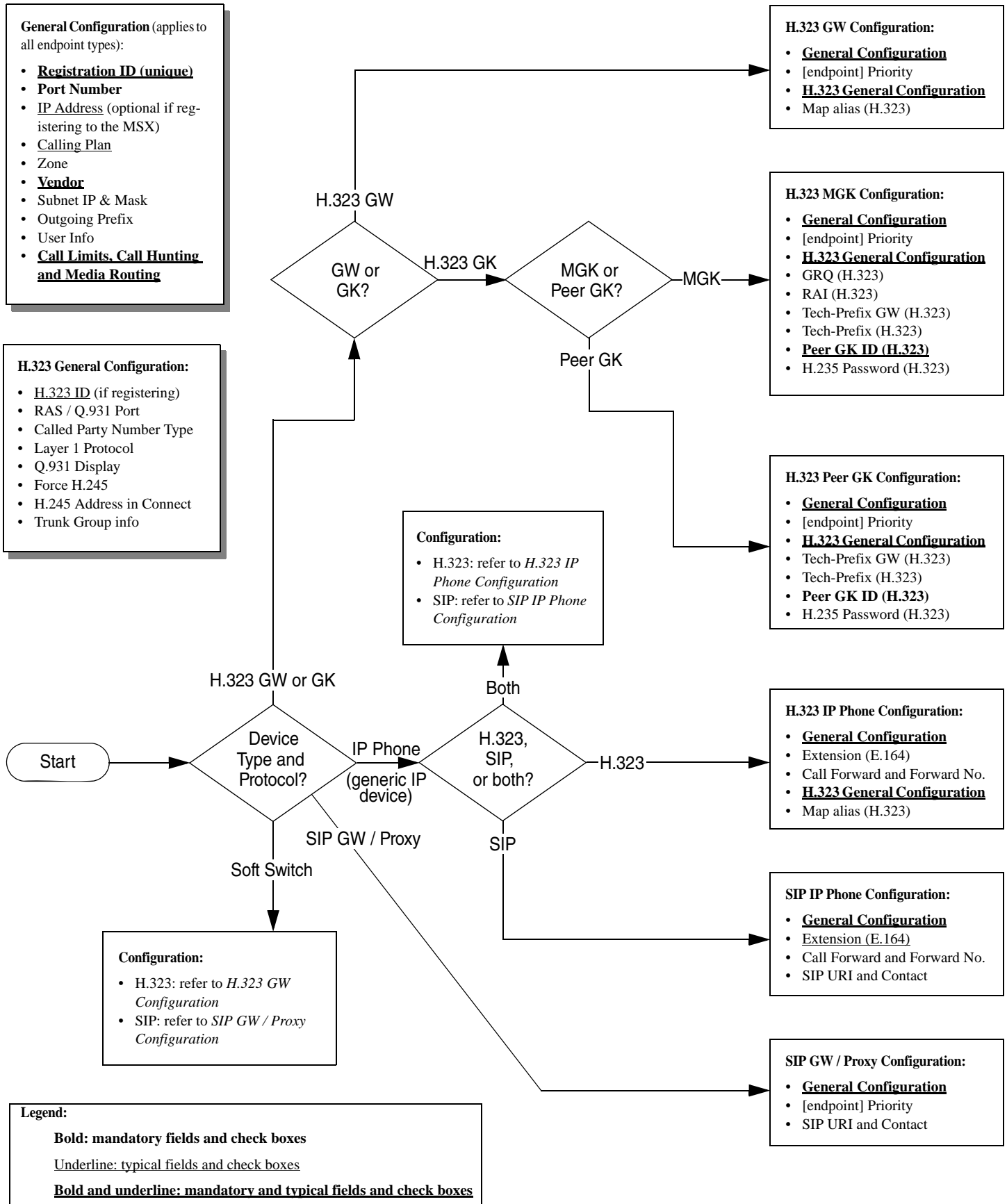
The MSX supports multiple ENUM servers (as endpoints defined with a *type* of *enum* in *cli*, or of Endpoint Type: ENUM Server in RSM Console; see *ENUM*

*server* on page 14.). It also supports overriding the global ENUM domain for each individual endpoint defined to the MSX.

The endpoint parameter for setting an ENUM domain for an endpoint is `enumdomain`. The command is:

```
cli iedge edit regid uport enumdomain domain string
```

# Adding an Endpoint



## MSX Administration

### Introduction

This chapter presents activities that an administrator will perform at various times to maintain the MSX system. Examples are adding users, system file backup, system restoration, etc. Each activity is covered in one of the following sections.

### Operating System maintenance

During the life of an MSX, it may become necessary for operating-system patches and various other OS-level upgrades to be applied, or other maintenance performed. If you believe there is a need to patch your NexTone Linux operating system, contact NexTone Support for assistance.

### Backing up an MSX

Use the steps in this section to back up the data defining your MSX. The subsections address system files and the endpoint/configuration database.

#### ***System file backup***

When backing up an MSX system, a complete copy of every file on the system disk (or even an image of the disk itself) can of course be taken. However, it may at times be preferable to merely take a snapshot of certain critical files that define targeted aspects of system configuration, without backing up the entire system as such. Such a backup file set could then be used to re-create a system from a clean MSX install.

In such cases, the files to capture are listed, by category, in Table 7. A few special cases are listed in the subsections below the table.



**Table 7. General MSX Backup Files**

Category	Files
Machine operating system/network configuration	/etc/passwd /etc/shadow /etc/system
Networking	/etc/hosts /etc/sysconfig/network /etc/resolv.conf /etc/ntp.conf /etc/localtime crontab files (/var/spool/cron/tabs/*)
Firewall	/etc/opt/ipf/ipf.conf /etc/opt/ipf/additional*
rsync and syslog	/etc/rsyncd.conf /etc/syslog.conf
MSX configuration	/opt/nextone/bin/h323gk*val /opt/nextone/bin/codemap* /opt/nextone/bin/mdevices.xml
Configuration files used by scripts under crontab	/opt/nextone/etc/dbback.cfg /opt/nextone/etc/cdrtrim.cfg /opt/nextone/etc/logpp.conf /opt/nextone/etc/events.conf
RSM (or NARS) Agent	/opt/narsagent/cdrstream*.xml /opt/narsagent/NARS.server.lc /opt/narsagent/nars.cfg /opt/narsagent/narslog.cfg
Output from cli db export	cli db export /opt/old-database gzip /opt/old-database
SNMP Configuration	/etc/snmp/snmpd.conf
CDRs and CDR formatting scripts	All files in /var/cdrs

**File-based ANI manipulation**

Text files used for this feature are stored in /opt/nextone/bin. The files are arbitrarily named, so you must either know them, or obtain them by executing the following commands:

```
cli cr list > ani.out
grep "src = /@" ani.out | more
```

The output from this command sequence will look like:

```
src = /@mex_1_1
src = /@gdl_1_1
src = /@gdl_1_1
```

The filenames are the strings following the “/@" in each line.

### **Configuration Database Backup**

To back up the endpoint configuration database:

1. Log on to the MSX as root.
2. Ensure that no one is updating endpoint configuration via CLI or RSM Console (so that you're not trying to export database contents that are being modified), then enter:

```
cli db export destination file path and name
```

This command extracts the database information and saves it in an editable XML format in the named file. Run this command periodically (or on-demand when needed, such as when upgrading the MSX software) to back up the MSX's endpoint configuration database.

## **Starting and stopping the MSX**

At times it may be necessary to start or stop the MSX processes, for example, when reorganizing the MSX's database.

**Note:** *When you stop and restart the iServer processes, in-progress calls are continued in some cases (SIP-SIP via SCM on redundant systems), and dropped in others (H.323 SIP-SIP calls in non-redundant systems). Calls that are partially set up (that is, calls that have not reached “steady-state”) are dropped in all cases.*

### **Determining MSX status**

To determine the status of the MSX, log onto the machine as root, and enter:

```
iserver all status
```

If the MSX is running, the output of this command shows the version numbers of four running processes (along with additional output<sup>1</sup>). For example:

```
NexTone iServer version-4.2c1-11, Wed Oct 18 19:37:52 EDT 2006
Copyright (c) 1998-2006 NexTone Communications, Inc.
```

If the MSX is *not* running, the output shows:

```
pm: No such process
rsd: No such process
gis: No such process
iServer not running
```

### **Stopping the MSX**

If the MSX is running, and you wish to stop it, log onto the machine as root, and enter:

```
iserver all stop
```

Some messages indicate that processes are being stopped, and the prompt returns. If the MSX processes are stopped on a non-redundant MSX system, in-process calls are dropped.

### **Starting the MSX**

If the MSX is not running, and you wish to bring it up, log onto the machine as root, and enter:

```
iserver all start
```

The machine responds with:

```
Nextone iServer is being started
```

## **Managing log files**

The nature of log files (system, error, H.323, CDR, etc.) is to grow until someone intervenes. The `logtrim` command provided to control this problem in releases prior to 4.0 is no longer needed. The Linux distribution on which the 4.0 and subsequent MSXs are based provides a `logrotate` command to accomplish the same end. Information on the `logrotate` command is widely available, including by `logrotate`'s man page.

Generally, the best way to control log file growth is to set up a cron job that runs `logrotate` periodically.

1. To just display the versions of installed modules, use `iserver all version`. Note that this command works even when the processes are not running, so it can't be used to determine system status.

## MSX Licenses

### Introduction

Access to MSX features is controlled by licenses. NexTone provides a license for each MSX. This license is stored as an XML element in the *servercfg* area and can be viewed, but not changed.

MSX licenses control the following software capabilities within your MSX platform:

- The number of virtual ports (vports) that can be used for concurrent calls
- The following MSX features:
  - H.323
  - SIP
  - SIP-T
  - SIP Registrar services
  - CAC (Call Admission Control)
  - FCE (Firewall Control Entity)
  - Media routing
  - RADIUS AAA services
  - SCM (Stateful Call Migration; SIP only)
  - NAT Traversal (SIP only)
  - NARS Agent support (separate license file)
  - QoS (quality of service) reporting in CDRs
  - Lawful Intercept
  - Emergency Call Admission Control
  - Signaling-Based DTMF and DTMF tone generation.
  - Rate Limiting
  - 3GPP Rx Interface: a license is required for both the Packet Cable Multi Media and policy server software

## License packs

### ***Vport-based licensing***

MSX licenses are based on a number of concurrent vports. To increase the number of concurrent calls that you can make in the network, you must purchase additional licenses.

### ***License error***

If you run out of licenses and have not purchased and installed additional licenses, the MSX limits what you can do, accordingly. Examples are:

- If you have too few vport licenses, calls will be refused, and the CDRs will reflect this.
- If you have not purchased a license for a feature controlled by an individual feature license (such as H.323, SIP, or FCE), that feature is disabled.

### ***License expiration warning***

If a license is within *one week* of expiring, CHECK LICENSE appears in black beneath the server's icon in the RSM Console Map window. If the license is within one *day* of expiration, that message appears in **red**.

Also, the **Alarms** window contains an entry if the license is near expiration (right-click the server name in the RSM Console Map window and choose Show Alarms to view that window.)

Double-click the CHECK LICENSE message beneath the MSX icon to bring up details about the license status.

## Obtaining an MSX license

When requesting a new or updated MSX license, ensure that you have the following information before contacting your NexTone Customer Support representative:

- **MAC Address.** Every Ethernet interface has a globally-unique identifier, called its MAC address. When requesting a license, you provide a MAC address from one of the interfaces on your MSX. While it can be a MAC address for any interface on the machine, the one used for MSX licensing should be one unlikely to change, such as the `eth0` interface. To obtain this identifier enter `ifconfig eth0` at the iServer command prompt.

The following gives a sample command output:

```
eth0      Link encap:Ethernet  HWaddr 00:0E:0C:31:DB:41
          inet addr:209.125.86.49  Bcast:209.125.86.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          ...
```

Record the data following `HWaddr` in the first line, which is the MAC address for `eth0`; in the example, `00:0E:0C:31:DB:41`.

If you are using a set of redundant MSXs, you must provide the host IDs of both servers.

- **Name and complete address of your organization**
- **Number of licenses required**
- **Release version number of MSX software** for which you need the license. Determine this by logging on to the MSX as `root`, and entering:

```
iserver all version
```

When you have this information ready, contact NexTone support in one of the following ways:

- E-mail [support@nextone.com](mailto:support@nextone.com)
- Access the web page at [www.nextone.com](http://www.nextone.com) and follow the “Support” link. Use your customer login ID and password to access NexTone support services.

## Installing an MSX license

**Caution:** *Before installing the new license file, copy your existing license file (`/usr/local/nextone/bin/iserverlc.xml`) to a separate location, saving it under a different name. This will ensure that you can revert to that file if you have problems using the new one.*

The MSX license is provided to you in an XML file named `<company-name>_<exp-date>_<hostid>.xml`. For example,

```
P4com_3-21-2004_2b3baf19.xml
```

Follow the steps given below to install this file on your system:

**Caution:** *To preserve its integrity, do not attempt to edit the license file. Even though it is human-readable, changing anything in it will cause it and the system to stop working.*

1. Log onto the system as `root`.
2. Change to the directory where the license file is stored:

```
cd /opt/nextone/bin
```

3. Copy the existing license file to a backup file by entering:

```
cp iserverlc.xml backup file name
```

4. Copy this file to the following MSX directory /opt/nextone/bin. The filename is expected to be iserverlc.xml. If the file has a different name, change it to iserverlc.xml.

*Note: If there is an iserverlc.xml file in this directory, replace it with the new file.*

5. Enter the following commands:

```
nxconfig.pl -l
```

## Preserving the license file via an MSX upgrade

When upgrading an MSX, the installation procedure utility asks you if you wish to use the license file from the previous version. Answer `yes` to continue using the existing license file.

## Viewing license status

To check the status of used and unused MSX licenses on your network, use the following steps:

1. Log on to the MSX as root.
2. At the command prompt, enter:

```
cli lstat
```

Below is an example of the output from this command:

License Node locked to	bda7fb16			
License Expiry Date	Fri Dec 31 00:00:00 2004			
Licensed Features	H323	SIP	FCE	RADIUS
Total VPORTS	100			
Available VPORTS	100			
Used VPORTS	0			
Total Media Routed VPORTS	100			
Available Media Routed VPORTS	100			
Used Media Routed VPORTS	0			

## Version compatibility

Note that between version iServer version 2.x and 3.x, the format of the license file changed. License files prior to MSX version 3 were in binary format, and named `iserver.lc`. License files version 3 and later are in XML format, and are named `iserverlc.xml`. These files are not interchangeable; changing the name of an older file to `iserverlc.xml` will result in a non-functioning system.

## QoS licensing

Quality of Service reporting is licensed at the following two levels:

1. *QoS (VLAN) Tagging*, which enables 802.1p/q priority and IP type-of-service (TOS) tagging. See *VLAN Support* on page 452.
2. *QoS Policing*, which enables RTP bandwidth policing, to limit bandwidth used by calls, as described in *Limiting bandwidth by call* on page 90.

## DTMF licensing

DTMF conversion includes four different combinations that affect how this feature is licensed:

1. Signaling-to-signaling DTMF conversion includes DTMF conversion between different signaling-based DTMF methods for SIP-to-SIP, H.323-to-H.323, and Inter-Working Function (IWF). No separate license key is necessary for this conversion. If you have licensed SIP, then SIP-to-SIP DTMF is available. If you have licensed SIP, H.323, and IWF, then all combinations of signaling-to-signaling DTMF conversion are available.
2. Signaling-to-media DTMF conversion includes SIP/H.323-based DTMF-to-RFC 2833 or G.711 DTMF support. A DTMFGEN license is required for this conversion. This license provides use of media-based DTMF tone generation.
3. Media-to-signaling DTMF conversion includes RFC 2833-based DTMF-to-SIP or H.323 DTMF support. There is no support for G.711 inband DTMF-to-signaling DTMF. DTMF\_DETECT for detecting different forms of media-based DTMF is required. This license only supports RFC 2833 DTMF detection.



4. Media-to-media DTMF conversion from media to signaling DTMF includes RFC 2833-based DTMF-to-G.711 inband DTMF. DTMF\_DETECT for detecting different forms of media-based DTMF is required. This license only supports RFC 2833 DTMF detection.

## The MSX Database

---

The MSX database is a repository of all MSX system configuration, endpoint, calling plan, and route information on the MSX. The database file is stored in binary format, but can be exported in XML format for backup or importation into other applications.

In redundant systems, a copy of the database resides on each machine, with one designated as the master, and the other as the slave. For more information on how this is laid out, see the section called *Database replication details* on page 156, in the *MSX Redundancy* chapter.

### Database administration

You can control and manage the MSX database using either of the following methods:

- Command Line Interface (CLI)
- RSM Console provisioning and configuration application (see RSM Console online help for detailed information on managing the database)

#### **CLI database commands**

When you first install the MSX, the database file is empty. You can populate the database with information on endpoints, calling plans, call routes, and bindings using RSM Console. Refer to RSM Console online help for detailed procedures to do this.

Once the database is populated by using RSM Console, you can perform any of the following tasks on your MSX database:

- Initialize the database
- Obtain information on the database
- Organize the database
- Switch databases
- Force a database backup
- Add information to an existing database

- Replace information in an existing database
- Delete selected information from the database
- Export the contents of the database for use by another application

### **Initializing the database**

A newly-created database needs to be initialized before it can be used. Please note that this should only be done when starting a fresh database.

To initialize the database, follow the steps given below.

1. Log on to the MSX as root.
2. Enter:

```
cli db init
```

Your database is initialized on running this command.

### **Obtaining information on the database**

*Note: DO NOT use this procedure when the MSX processes are running.*

To obtain information on the size of the database, follow the steps given below.

1. Log on to the MSX as root.
2. Enter:

```
cli db info
```

A display similar to this appears on the screen:

```
Current Database:      msw

Provisioning Tables

endpoints:             1 Entries
endpoints_dynInfo:     1 Entries
callingplans:          0 Entries
callingroutes:         0 Entries
cpbindings:            0 Entries
vnet:                  0 Entries
realms:                0 Entries
iedgegroups:           0 Entries
igrp_dynInfo:          0 Entries
subnets:              0 Entries
fax:                   0 Entries
cdcprofiles:           0 Entries
triggers:              0 Entries

Other Tables

MSW_Journal:           0 Entries
```

```
event: 1 Entries
alarm_info: 2 Entries
msw_status: 93 Entries
alloc_stats: 10 Entries
route_query: 0 Entries
call_stats: 144 Entries
sip_stats: 10 Entries
clihistory: 1 Entries
```

### **Organizing the database**

To optimize disk space, you can compact the database such that all empty records are packed. This procedure may however, not always result in a reduction of database size.

1. Log on to the MSX using the root account.

**Caution:** *Before reorganizing the database with the `org` command, make sure the MSX processes are stopped. See Starting and stopping the MSX on page 98.*

2. Enter:

```
cli db org
```

Running this command re-organizes the database to optimize disk space.

### **Switching databases**

The MSX provides the ability to switch databases during operation, via the `cli db switch` command. The end result of the switch command is the same as for `cli db copy`, which copies a new database on top of an existing one, but the copy method can result in active calls being dropped, and new calls to some endpoints not being set up.

The switch database command makes the transition to the new database seamless and graceful. Calls are not dropped, including existing calls to endpoints that were in the old database and not in the new database, and valid CDRs are generated for all calls.

This “switch” methodology can be typically used to make large changes to databases in a graceful fashion. Note, however, that the “switch” methodology takes about 5-10 times longer than the “copy” method to import the elements of the database into memory. This information should be taken into consideration during network design and configuration changes.

The command syntax is as follows:

```
cli db create newdb
where newdb contains the new database information in XML format.

cli db switch newdb
```

where `newdb` is the database (represented by files with `.gdbm` suffix) created by the first step.

### **Adding information to an existing database**

To bulk-add information from an XML file to an existing database, follow the procedure given below.

1. Log on to the MSX as root.
2. Enter:

```
cli db add name of your XML file
```

On incorporating contents of the XML file, if the MSX finds records that already exist in the database, it reports them as errors.

**Caution:** *Before reorganizing the database with the `org` command, make sure the MSX processes are stopped. See Starting and stopping the MSX on page 98.*

3. If desired, re-organize the database by entering:

```
cli db org
```

Running this command compacts the database, and is recommended, but not required, after adding a large number of new records.

### **Replacing information in an existing database**

If you have an existing database and want to replace some of its contents with information in a different XML file, follow the procedure given below.

1. Log on to the MSX as root.
2. Enter:

```
cli db replace name of your XML file
```

Running this command replaces all common information between the two files, with the information in the XML file you imported to the database. Before replacing the entries in the database, the MSX matches the calling plan names, call route names, and registration ID of endpoints. It only replaces the entry if these match. If you wish to replace call bindings, then the calling plan name and call route name must be an exact match between the XML file and the existing database.

**Deleting information from an existing database**

If you wish to delete the contents stored in an XML file from the database, follow the procedure given below.

*Note: The MSX does NOT compare the information in the two files before deleting them.*

1. Log on to the MSX as root.
2. Enter:

```
cli db delete name of your XML file
```

On running this command, the MSX deletes all common information between the two files, from the database.

**Caution:** *Before reorganizing the database with the `org` command, make sure the MSX processes are stopped. See Starting and stopping the MSX on page 98.*

3. If desired, re-organize the database by entering:

```
cli db org
```

Running this command compacts the database, and is recommended, but not required, after deleting a large number of records.

**Database export and import considerations**

As covered in the next sections, there are occasions when exporting and importing database contents may be useful for transferring, copying or archiving data. It can also be helpful sometimes to put data into a test or lab machine, when troubleshooting an issue.

Data moved using the export and import functions is placed into an editable text file containing XML data. Please note the following:

- Port numbers in the XML file are the numbers internal to the database, and are not the same as the ones shown in RSM Console. In the database, port numbering begins with 0 (zero), but RSM Console begins at 1 (one). Therefore, when correlating them, add 1 to the database port to obtain the number that RSM Console shows.
- **Important:** When exporting data to import it into a database for troubleshooting (i.e., problem re-creation) purposes, be sure to remove all instances of H.323 Master Gatekeepers (sgatekeepers), by editing the contents of the resulting XML file. Failure to observe this may cause registration problems if two MSXs try to register using the same H.323 ID, to the same gatekeeper.

## Manually upgrading the database

On upgrading your MSX software, the database is automatically upgraded. However, if you wish to manually upgrade your database, follow the procedure given below and run the three CLI database commands in the order they are listed.

1. Log on to the MSX as root.
2. Ensure that no one is updating endpoint configuration via CLI or RSM Console (so that you're not trying to export changing database content), then enter:

```
cli db export path to destination file
```

This command extracts the database information and saves it in an editable XML format in the specified file. It's good to run this command periodically to back up the MSX database.

**Note:** *You can also use the `cli db export filename` command to back up your database in an editable format, or to incorporate a file in the proper format into an already existing database.*

3. Format the information into the proper XML format for the database by entering:

```
cli db create filename from Step 2
```

4. Copy the information into the new database, and put it online by entering:

```
cli db copy filename from Step 2
```

## Example manual database upgrade

A manual database upgrade looks like this example:

```
cli db export /var/tmp/exporttest
cli db create /var/tmp/exporttest
cli db copy /var/tmp/exporttest
PartitionId for all records will be imported as admin, Do you
want to continue?[y|n]
```

To continue, enter: `y`. the process finishes with:

```
Importing...
Schemaname is _var_tmp_exporttest
SCHEMA _var_tmp_exporttest created successfully
```

## Replacing an existing database

To replacing an existing database, follow the steps given below.

1. Log on to the MSX as root.

**Caution:** *Before reorganizing the database with the `org` command, make sure the MSX processes are stopped. See Starting and stopping the MSX on page 98.*

2. Export the current database and create a backup in case problems arise and a fallback is necessary by typing:

```
cli db export path to destination file
cli db org
```

3. Create and import the new database file in the required format by typing:

```
cli db create complete path with new XML file name
```

**Note:** *If you see an error message regarding the XML file during this step, adjust the file accordingly and rerun this command until all errors are cleared.*

4. Stop the MSX by typing:

```
iserver all stop
```

5. Erase the old database by typing:

```
cli clean db all
cli db init
```

6. Ensure that shared memory is released by typing:

```
ipcrmall
```

7. Copy the new database file to the appropriate location by typing:

```
cli db copy same path and file name as in Step 3
```

8. Start the MSX processes by typing:

```
iserver all start
```

9. Wait ten seconds and verify that all MSX processes (`gis`, `execd`, `pm`, `java` and on redundant cluster machines, `rsd`) are running by typing:

```
iserver all status
```



## Performance Tuning and Statistics

### Configuring max calls

The `h323-maxcalls` *servercfg* attribute defines the number of legs in a call, so the actual number of calls is half `h323-maxcalls`' value. Modify this attribute as detailed below.

1. Compute the max calls using the following formula:  
Number of max calls =  $2.5 \times (\text{number of vports})$
2. Log on to the iServer as `root`.
3. Set the `h323-maxcalls` attribute to the calculated value by entering:  

```
nxconfig.pl -e h323-maxcalls -v value
```

where value is the value you calculated in step 1.

### Patch levels required for optimization

Installing certain patches may help optimize your system performance. For specific patch information, contact NexTone Support.

### Memory requirements

To compute the amount of RAM required to optimize your system, use the following formula:

Total RAM = (Number of H.323 Call Legs x 100K) + (Number of SIP Call Legs x 30K) + (Shared Memory)

To obtain the Shared Memory value, enter:

```
cat /proc/sys/kernel/shmmax
```

This returns the maximum shared memory the iServer can use, in bytes.

## Statistics and monitoring

Statistical data is useful when monitoring the health of your system. The following types of statistics are automatically compiled in the system and can be retrieved on demand:

- system statistics
- endpoint statistics

These statistics can be accessed using the iServer's Command Line Interface (CLI), as shown in the next sections.

### ***Getting system statistics***

The `cli stats` command (see page B-11) provides peg counts of H.323 setup and connect messages over the last 10 second, 10 minute and 10 hour periods that the iServer has been in continuous operation. To use this command, simply enter:

```
cli stats
```

### **Interpreting cli stats output**

The output from the `cli stats` command is as follows:

- `H323StackQ` is the number of events pending in the H.323 stack's queue
- `ThreadPoolQ` is the number of events pending in the application's queue
- `Active Calls` is the number of currently active calls in the system as of the moment the snapshot was taken (i.e., when the command was entered).
- `Setups` is the number of H.323 setup messages received in a second/minute/hour for the last ten seconds/minutes/hours, with the most recent second/minute/hour shown on the right.
- `Connect` is the number of H.323 connect messages received in each of the last ten seconds/minutes/hours, with the most recent second/minute/hour on the left.
- `outSetups` is the number of H.323 setup messages transmitted in each of the last ten seconds/minutes/hours, with the most recent second/minute/hour on the left.

### **Statistics for SIP call processing**

The `cli stats` command also supports an option to show statistics for SIP call processing. The command is:

```
cli stats sip
```

In addition to the call setup and connection information shown for the command without the `sip` option, the output provides the number of INVITE's, 1xx, 2xx, 3xx and ACK's and BYE's. As for the command with no option, the statistics are for the preceding 10 seconds, 10 minutes and 10 hours.

### **Getting endpoint statistics**

To retrieve endpoint statistics, type the following command

```
cli iedge lkup regid uport
```

This command displays the following statistics:

- Last time endpoint became unregistered (inactive), if the endpoint is registering with the gatekeeper
- Last time maxcalls was reached on that endpoint or an RAI was received with the parameter “almost out of resources” set to True.
- Last time an outgoing call was attempted on the gateway, but did not go through because the gateway had reached max calls or sent out an RAI with the parameter “almost out of resources” set to True.

## SNMP Support

### Introduction

SNMP, the *Simple Network Management Protocol*, provides a standardized way to monitor and manage SNMP-capable devices on a network using MIBs, or *Management Information Bases*. Traps use available MIB *objects* to formulate meaningful diagnostic messages, which are then forwarded to one or more management stations. SNMP `get` operations can also be used by a management station to retrieve MIB information. At this time, only passive and active monitoring are available for the MSX. Remote control monitoring is not available.

### SNMP support features

MSX SNMP support is based on the open source Net-SNMP package. It supports a full-featured, extensible SNMP agent. NexTone provides the following Net-SNMP support:

- Linux iServer support
- Feature license not required
- SNMPv1 and SNMPv2c support
- SNMPv3 security support
- BMC hardware sensors with chassis-related MIBs support
- DOS attack statistics support (NexTone MIB)
- IP-MIB and IP-FORWARD-MIB support

You must first configure the iServer SNMP agent to monitor the health of the operating system and applications. SNMP support lets you monitor one or more servers remotely by using an SNMP-compliant network management system (NMS) or other management software to receive traps. Traps are SNMP notifications, sent over the network to specific NMS hosts, when pre-configured events occur. The configured SNMP agent generates traps for operating system events such as disk space warnings and authentication failures, as well as NexTone proprietary notifications.

The NexTone Net-SNMP package is not configured to generate SNMP informs.

### **Net-SNMP Security Packages**

The standard Net-SNMP package provided by NexTone uses SNMPv3 security for authentication and authorization. Because of this you must configure correct authentication details (user, password, encryption method, etc.) in both the NexTone server and the NMS that will receive traps sent to it by the MSX.

*Note: If SNMPv3 security is not required, you can configure the NexTone SNMP agent to use the simpler SNMPv2c security model.*

## **Configuration**

Before the SNMP service is started, the Net-SNMP agent must be configured. Your NMS should recognize the NexTone traps, as well as the traps sent by the agent. Aside from the standard SNMPv2c MIBs and the SNMPv3 security MIBs, the following MIBs should be loaded into the NMS:

- DISMAN-EVENT-MIB
- ENTITY-MIB
- ENTITY-SENSOR-MIB
- HOST-RESOURCES-MIB
- IF-MIB
- IP-FORWARD-MIB
- NET-SNMP-AGENT-MIB
- NETWORK-SERVICES-MIB (for applTable support)
- NEXTONE-SMI-MIB
- NEXTONE-NOTIFICATION-MIB
- NEXTONE-IP-LAYER-RATELIMITING-MIB
- NEXTONE-SESSION-LAYER-RATELIMITING-MIB
- NEXTONE-CALLTAP-MIB
- UCD-SNMP-MIB

All of the above MIB files are located in the `/usr/share/snmp/mibs` directory.

*Note: Be sure to log in as root to execute the steps in the following sections.*

### **The SNMP configuration utility**

NexTone provides a configuration utility program for the SNMP agent. It includes the following four modules:

- *User Management*

- *NMS Management*
- *Agent Management*
- *Heartbeat Management*
- *Threshold Management*

To use SNMP, you must run the User Management, NMS Management, and Agent Management modules. The Heartbeat and Threshold Management modules appearing in the main menu are optional.

1. Start the SNMP Configuration utility by entering:

```
nexsnmp-config
```

The main menu displays:

```
NexTone SNMP Configuration
-----
1) User Management
2) NMS Management
3) Agent Management
4) Heartbeat Management
-----
5) Threshold Management
-----
s) Save and Quit
q) Discard and Quit
-----
Select [1-5,s,q]:
```

**Note:** *To exit the utility from a sub-menu at any time, without saving your changes, press <Ctrl>+C.*

2. Type 1 after the Select prompt to run the User Management module. The Username prompt displays:

```
** Username must be at least 6 characters **
Username:
```

3. Type a user name after the Username prompt and press <Enter>. The User Access menu displays:

```
User Access:
1) ro
2) rw
Select [1-2]:
```

4. Type a 1 or 2, using the following criteria to decide which option is appropriate for your use and press <Enter>:

- `ro` (option 1) grants the user read-only access. Users can only issue commands that read existing SNMP data, such as `snmpget` and `snmpwalk`. Use this option if you are uncomfortable with having write access.
- `rw` (option 2) grants read and write access. Read-write authorized users can alter data and settings, such as can be done with the `snmpset` command. Currently NexTone SNMP MIBs do not support write operations.

The SNMP Version menu displays after you have selected the type of user access.

5. Type the number of the SNMP version you are using after the Select prompt and press <Enter>.

SNMP Version:

- 1) 1
- 2) 2c
- 3) 3

Select [1-3]:

If you choose either version 1 or version 2c, you must provide a Community Name.

Press <Enter> to keep the current name (by default, the name is `public`), or type a different name, and press <Enter>. Skip to step 10, below when finished.

6. If in step 5 you chose SNMP version 3, you must select an authentication type from the Authentication Type menu:

Authentication Type:

- 1) MD5
- 2) SHA

Select [1,2]:

Type a 1 or a 2, depending on your authentication encryption type, and press <Enter>. The Authentication Passphrase prompt displays.

7. Authentication types require a passphrase for execution. Enter a passphrase for the authentication type you chose after the Passphrase for prompt and press <Enter>:

\*\* Passphrase must be at least 8 characters \*\*

Passphrase for <auth-type>:

Confirm the passphrase when prompted. The Authorization Type menu displays.

8. Choose an authorization type from the Authorization Type menu. Type a 1 or 2 after the Select prompt depending on your authorization encryption type and press <Enter>.

```
Authorization Type:
1)DES
2)AES
Select [1,2]:
```

9. Authorization types require a passphrase for execution. Enter a passphrase for the authentication type you chose after the Passphrase for prompt and press <Enter>:

```
** Passphrase must be at least 8 characters **
Passphrase for <auth-type>:
```

Confirm the passphrase when prompted. The main menu displays.

10. Type 2 after the Select prompt to run the NMS Management module from the main menu and to add IP addresses for the remote NMS servers at your site.

If you have not previously configured remote NMS servers at your site, the following menu appears:

```
NMS Management
-----
```

```
Configured NMS{s}:
None
```

```
Select (a)dd or (r)eturn:
```

- a. Type r after the Select prompt to exit the NMS Management menu and return to the main menu. If you enter an a, the program displays the following prompt:

```
** Enter a blank line to stop adding addresses **
```

```
Remote NMS [0] IP:
```

- 10.1 Type an IP address after the Remote NMS prompt and press <Enter> to enter the first remote NMS server IP address. You can enter multiple remote NMS server IP addresses after the subsequent prompts that appear.



- 10.2 After adding the last address, press <Enter> twice. A menu listing the IP addresses that you entered for the remote NMS servers at your site appears. The following menu also appears if you have previously configured one or multiple remote NMS servers.

```
NMS Management
```

```
-----
```

```
Configured NMS{s}:
```

```
1) 10.1.4.30
```

```
2) 11.11.0.20
```

```
Select (a)dd, (d)elete, d(e)lete all, or (r)eturn:
```

- 10.3 Type `r` after the Select prompt to exit NMS Management and to return to the main menu. Enter an `e` to delete any or all of the listed destination IP addresses. If you enter a `d`, the following prompt displays:

```
Select [1-2]:
```

```
*****
```

Type the number after the Select prompt of each destination IP address to delete, then press <Enter>. When finished, type `r` to return to the main menu.

11. At the main menu, type `3` after the Select prompt to run the Agent Management module. This module configures the SNMP agent IP from which traps will be sent, and provides the computing `engineID` to use when creating a user on the remote NMS. The agent IP address should be the same address assigned to the iServer management interface.
- a. If you have already assigned the iServer management interface IP address (the `mgmt_ip` attribute in `servercfg`; see *Global configuration using nxconfig.pl*, on page 18), the following prompt displays:

Agent Management  
-----

Management IP is configured, use it as agent IP

\*\* IP change will cause the engineID to be re-generated \*\*  
SNMP Agent IP [<mgmt ip>]:

Press <Enter> to keep the current agent IP address. You can change it by entering a new address. Proceed to step 13 when finished.

- 11.1 If you previously assigned an agent IP address by running `nexsnmp-config`, the following prompt indicates that the agent IP has already been configured. You can still change it:

Agent Management  
-----

Agent IP is already configured

\*\* IP change will cause the engineID to be re-generated \*\*  
SNMP Agent IP [<agent ip>]:

Press <Enter> to keep the current agent IP address. You can change it by entering a new address, and then proceed to step 12.

- 11.2 If you have not previously configured the management interface IP address in `servercfg`, the following prompt appears:

Agent Management  
-----

Agent IP is not configured

\*\* IP change will cause the engineID to be re-generated \*\*  
SNMP Agent IP []:

Type the agent IP address after the SNMP Agent IP prompt and press <Enter>. The IP address that you enter must be already configured on the iServer at your site.

12. An engineID (a 64-bit hexadecimal number) is computed, based on the local host MAC address, plus a 4-bit random number. You will need this computed engineID when creating the user account on the remote NMS. The following information displays:

```
Agent IP: <agent IP>
engineID: 0x0102030405060708
```

```
Use the above engineID when creating user on remote NMS
Press <Enter> to continue
```

Record the engineID (we recommend copy-and-paste to a text file), and press <Enter> to continue. The main menu displays.

13. At the main menu type 4 after the Select prompt to run the optional Heartbeat Management module. The Heartbeat Management configuration screen displays:

```
Heartbeat Management
-----
```

```
** Enter 0 (zero) second to disable heartbeat monitor **
```

```
Heartbeat Rate (0 sec):
```

14. Type a value for the heartbeat rate to monitor and press <Enter>. The heartbeat message sent by the SNMP agent is "Heartbeat". The main menu displays.

15. At the main menu, type 5 after the Select prompt to run the optional Threshold Management module. The Threshold Management menu displays:

```
Threshold Management
-----
```

```
1) Ethernet Link Monitor
2) Free Disk Space Monitor
3) Memory Usage Monitor
4) Idle CPU Monitor
```

```
Select [1-4, r]:
```

- a. Type 1 after the Select prompt to change the link status monitor. The Interval prompt displays:

```
Interval [10]:
```

Type the interval, in seconds, after the Interval prompt. The value that you enter is the time in seconds at which the link status will be checked. To keep the existing internal value, press <Enter>. The default interval is 10 seconds.

- 15.1 Type 2 after the Select prompt to change the disk space monitor for a specified partition. The Disk Space Monitor menu displays:

```
Threshold(s)::disk (byte or %)
1) / 10%
2) /var 10%
Select (a)dd, (d)elete or (f)inish:
```

**To add a new disk space threshold:** Type a after the Select prompt and press <Enter>. The following prompt displays:

```
Threshold (partition [low] high):
```

where `partition` is the disk partition to monitor, and `low` and `high` are the minimum and maximum values of free space for this partition that will trigger a trap.

Specify each limit as either a number of bytes (by entering an integer), or a percentage of the total partition space (entered as an integer from zero to 100 with a percent sign appended, for example, 30%). If only one number is given, it becomes the high limit, and the low limit is set to 0. If a low limit is entered, both high and low limits must be of the same form (number of bytes or percentage).

**Note:** *Multiple thresholds can be defined for one partition.*

**To delete a disk space threshold:** Type d. after the Select prompt and press <Enter> the following prompt appears:

```
Select [1-n]:
```

Type the number of the disk space threshold to delete after the Select prompt and press <Enter>.

When you finish adjusting disk space thresholds, type f to exit the Disk Space Threshold menu.

- 15.2 Type 3 after the Select prompt to change the memory utilization monitor. The following menu appears:

```
Threshold(s)::memory (kbyte)
1) gis 2700000

Select (a)dd, (d)elete or (f)inish
```

**To add a new memory utilization threshold:** Type a. after the Select prompt and the following prompt appears:

```
Threshold ([proc name] low [high]):
```

where `proc_name` is the process name to monitor. This can be any process name returned by a `ps` command. If a `proc_name` value is not supplied, system memory utilization is monitored. `low` and `high` are the minimum and maximum thresholds for memory usage by the specified process or the system. The value is specified in Kilobytes. If only one integer value is specified, it is used as the `low` value and no maximum memory threshold is set.

Memory utilization thresholds can be defined for specific processes or for the system as a whole. Multiple thresholds can be defined for either specific processes or for the system as a whole.

**To delete a memory utilization threshold:** Type d. after the Select prompt and the following prompt appears:

```
Select [1-n]:
```

Enter the number of the memory utilization monitor to delete.

When you finish adjusting memory utilization thresholds, type f to exit the menu.

- 15.3 Type 4 after the Select prompt to change CPU utilization monitor thresholds. Note that multiple CPU utilization thresholds can be defined. Ranges you define here represent unacceptable CPU idle percentage ranges. CPU idle percentages in these ranges will trigger a trap.

The CPU Utilization Monitor appears:

```
Threshold(s)::cpu (%)
```

```
1) 20%
```

```
Select (a)dd, (d)elete or (f)inish
```

**To add a new CPU utilization threshold:** Type *a.* after the Select prompt and the following prompt appears:

```
Threshold ([low] high):
```

where *low* and *high* are the minimum and maximum thresholds for CPU usage. If only one integer value is specified, it is used as the *high* value and the *low* value is set to 0.

**To delete a CPU utilization threshold:** Type *d.* after the Select prompt and the following prompt appears:

```
Select [1-n]:
```

Enter the number of the CPU utilization monitor to delete.

When you finish adjusting CPU utilization thresholds, type *f* to exit the CPU Utilization Threshold menu. The Threshold Management menu reappears.

15.4 Type *r* to exit the Threshold Management menu. The main menu displays.

16. At the main menu, type *s* after the Select prompt to save your changes and restart SNMP, or *q* to exit without saving the new configuration.

## System route configuration

Trap messages are sent through the MSX management interface. Because of this, the MSX must be on a network that is routable to your remote NMS.

## Controlling agent operation

The SNMP service can be stopped or restarted any time by specifying either *stop* or *restart* in the following command string:

```
/etc/init.d/snmpd start
```

A restart (or `kill -HUP` sent to the `snmpd` process) puts configuration changes into effect. You can obtain the SNMP service's status with the `status` command. The SNMP agent is configured to start in run levels 2, 3 and 5 at boot time.

## Available traps

### **SNMP Agent**

The Net-SNMP agent generates the traps listed in Table 8.

All agent traps related to hardware and operating system events are generated according to the `DISMAN-EVENT-MIB` specification. The `DISMAN-EVENT-MIB` is a product of the IETF Distributed Management Working Group and is described in RFC 2981.

`DISMAN-EVENT-MIB` notifications are based on triggers that the SNMP administrator configures in the agent. Triggers can be based on thresholds or events monitored by the Net-SNMP agent running on the MSX. When a `DISMAN-EVENT-MIB` trap is generated, several objects, including the one being monitored, are included in the trap.

NexTone has predefined several useful triggers that generate `DISMAN` notifications or traps. Table 8 lists each trap and the SNMP objects included.

Table 8. Net-SNMP Agent Traps

Event Description	Cause	Recommended Response	SNMP Trap Objects and Example Values
Free space on a monitored partition is within a threshold range.	<ul style="list-style-type: none"> <li>Log rollover/trim functions have failed</li> <li>Excessive core dump files generated</li> <li>CDR file growth off-loaded.</li> </ul>	<p>Check the trap output to determine which partition is over limit.</p> <p>If CDR files are the cause, move old files to another partition or offline. See <i>Setting a rule for opening new CDR log files</i> on page 427 for more info on CDR files.</p>	DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (3022) 0:00:30.22
			SNMPv2-MIB::snmpTrapOID.0 = OID: DISMAN-EVENT-MIB::mteTriggerFired
			DISMAN-EVENT-MIB::mteHotTrigger = STRING: Disk Space Low
			DISMAN-EVENT-MIB::mteHotTargetName = STRING:
			DISMAN-EVENT-MIB::mteHotContextName = STRING:
			DISMAN-EVENT-MIB::mteHotOID = OID: UCD-SNMP-MIB::dskErrorFlag.1
			DISMAN-EVENT-MIB::mteHotValue = INTEGER: 1
			UCD-SNMP-MIB::dskPath.1 = STRING: /
			UCD-SNMP-MIB::dskErrorMsg.1 = STRING: / 91% disk space free [90%, 95%)
<p>Ethernet link state has changed.</p> <p><i>Note: Enabling this function could generate excessive traps in HA systems</i></p>	<ul style="list-style-type: none"> <li>Disconnected cable</li> <li>HA system failover</li> <li>Device at other end of link went down</li> <li>Manual stop</li> </ul>	<p>If trap indicates that link state has changed to “down”, check cabling and attached device for physical connection.</p>	DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (6318) 0:01:03.18
			SNMPv2-MIB::snmpTrapOID.0 = OID: DISMAN-EVENT-MIB::mteTriggerFired
			DISMAN-EVENT-MIB::mteHotTrigger = STRING: Generate linkUp
			DISMAN-EVENT-MIB::mteHotTargetName = STRING:
			DISMAN-EVENT-MIB::mteHotContextName = STRING:
			DISMAN-EVENT-MIB::mteHotOID = OID: IF-MIB::ifOperStatus.2
			DISMAN-EVENT-MIB::mteHotValue = INTEGER: 1
			IF-MIB::ifDescr.2 = STRING: e1000g0



Table 8. Net-SNMP Agent Traps

Event Description	Cause	Recommended Response	SNMP Trap Objects and Example Values
Size of memory allocated to a running process (or the whole system) falls in a threshold range.	Generally associated with a leak in the main call routing process (gis).  <i>Note: Correct configuration depends on installation; see iServer release notes.</i>	Log on the system. Use <code>ps</code> command to locate the PID of the process. Analyze to determine cause of excessive memory, if possible. Kill or restart controlling application.	DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (11333) 0:01:53.33
			SNMPv2-MIB::snmpTrapOID.0 = OID: DISMAN-EVENT-MIB::mteTriggerFired
			DISMAN-EVENT-MIB::mteHotTrigger = STRING: High Memory Usage
			DISMAN-EVENT-MIB::mteHotTargetName = STRING:
			DISMAN-EVENT-MIB::mteHotContextName = STRING:
			DISMAN-EVENT-MIB::mteHotOID = OID: UCD-SNMP-MIB::memSwapError.10
			DISMAN-EVENT-MIB::mteHotValue = INTEGER: 1
			UCD-SNMP-MIB::memSwapErrorMsg.10 = STRING: snmpd: 5080kb memory used [5000kb, 6000kb) Or UCD-SNMP-MIB::memSwapErrorMsg.10 = STRING: SYSTEM: 5080kb memory used [5000kb, 6000kb)"
CPU idle average has fallen within a threshold range	High incoming call volume, DOS attack	Check endpoints for call volume, check logs for invalid messages (DOS).	Note that on multiprocessor systems, threshold monitoring is based on an overall total of utilization for all processors in an iServer, rather than individual CPUs.
			DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (5031) 0:00:50.31
			SNMPv2-MIB::snmpTrapOID.0 = OID: DISMAN-EVENT-MIB::mteTriggerFired
			DISMAN-EVENT-MIB::mteHotTrigger = STRING: High CPU Usage
			DISMAN-EVENT-MIB::mteHotTargetName = STRING:
			DISMAN-EVENT-MIB::mteHotContextName = STRING:
			DISMAN-EVENT-MIB::mteHotOID = OID: UCD-SNMP-MIB::ssErrorFlag.1
			DISMAN-EVENT-MIB::mteHotValue = INTEGER: 1
			UCD-SNMP-MIB::ssErrorMessage.1 = STRING: CPU: 19.84% idle (0%, 20%)

**Table 8. Net-SNMP Agent Traps**

Event Description	Cause	Recommended Response	SNMP Trap Objects and Example Values
SNMP agent startup	Normal startup	None	DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (20) 0:00:00.20
			SNMPv2-MIB::snmpTrapOID.0 = OID: SNMPv2-MIB::coldStart
			SNMPv2-MIB::snmpTrapEnterprise.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
SNMP agent shutdown	Normal shutdown	None	DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (620) 0:00:06.20
			SNMPv2-MIB::snmpTrapOID.0 = OID: NET-SNMP-AGENT-MIB::nsNotifyShutdown
SNMPv3 authentication failure	Unauthorized access or incorrect v3 configuration.	If authorized, verify username and password.	DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (4108) 0:00:41.08
			SNMPv2-MIB::snmpTrapOID.0 = OID: SNMPv2-MIB::authenticationFailure
			SNMPv2-MIB::snmpTrapEnterprise.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10

## Application

The NexTone software provides an SNMP service that generates traps in response to critical or serious application events. It does *not* provide a view into any managed objects. As such, this application is categorized as a *notification originator* according to the definition in RFC 3413.

To decode the NexTone traps in your NMS, load the NEXTONE-SMI-MIB.txt and NEXTONE-NOTIFICATION-MIB.txt files, located in the MIBS directory, /usr/share/snmp/mibs

The NexTone SNMP enterprise Object Identifier (OID) is:

```
iso(1) org(3) dod(6) internet(1) private(4) enterprises(1)
nextone(7684)
```

Application traps are listed in Table 9. For more-detailed response recommendations, refer to Table 10, *Events, Causes and Recommended Responses*, on Page 137.

Note that in the descriptions that follow, *HA* means a *high-availability* configuration consisting of two iServers, where at a given time, one host is active and the other is a standby. A *failover* is the event when the two switch their respective roles.

**Table 9. NexTone Application Traps**

Event Description	Cause	Recommended Response	SNMP Trap Objects and Example Values
Call routing service is available	Normal startup	None	SNMPv2-MIB::snmpTrapOID.0 = OID: NEXTONE-NOTIFICATION-MIB::callRoutingServiceStatus NETWORK-SERVICES-MIB::applOperStatus = 1 (up) NETWORK-SERVICES-MIB::applName = <proc name>
Host is in standby mode	The host gets back-up/standby status in HA configuration	None	SNMPv2-MIB::snmpTrapOID.0 = OID: NEXTONE-NOTIFICATION-MIB::haStatus NEXTONE-NOTIFICATION-MIB::nexToneHaState.0=3(haStandby)
Host is in primary mode	The host gets primary/active status in HA configuration	None	SNMPv2-MIB::snmpTrapOID.0 = OID: NEXTONE-NOTIFICATION-MIB::haStatus NEXTONE-NOTIFICATION-MIB::nexToneHaState.0=2(haPrimary)
Failover has occurred	Following a failover in HA configuration, the new primary sends this.	Determine cause and bring host back online.	New primary sends: SNMPv2-MIB::snmpTrapOID.0 = OID: NEXTONE-NOTIFICATION-MIB::haStatus NEXTONE-NOTIFICATION-MIB::nexToneHaState.0=2(haPrimary)
Internal error - service restarted	iServer process error	Check iServer log.	SNMPv2-MIB::snmpTrapOID.0 = OID: NEXTONE-NOTIFICATION-MIB::callRoutingServiceStatus NETWORK-SERVICES-MIB::applOperStatus =5 (restarting) NETWORK-SERVICES-MIB::applName=<proc-name>  This will be followed by a callRoutingServiceStatus trap indicating the process is up if restarting was successful.
Call routing service not responding	GIS is hung <sup>a</sup>	Check iServer logs for last operations.	SNMPv2-MIB::snmpTrapOID.0 = OID: NEXTONE-NOTIFICATION-MIB::callRoutingServiceStatus NETWORK-SERVICES-MIB::applOperStatus =4 (congested) NETWORK-SERVICES-MIB::applName=<proc-name>
Unresponsive call routing process is being dumped	GIS is hung - PM is asking it to dump memory for debugging	Contact NexTone support. Save gcore file for analysis.	Call routing service not responding event followed by: SNMPv2-MIB::snmpTrapOID.0 = OID: NEXTONE-NOTIFICATION-MIB::callRoutingServiceStatus NETWORK-SERVICES-MIB::applOperStatus =2 (down) NETWORK-SERVICES-MIB::applName=<proc-name>

**Table 9. NexTone Application Traps**

Event Description	Cause	Recommended Response	SNMP Trap Objects and Example Values
Call routing service is stopping	GIS is being shutdown by PM.	Check PM log	SNMPv2-MIB::snmpTrapOID.0 = OID: NEXTONE-NOTIFICATION-MIB::callRoutingServiceStatus NETWORK-SERVICES-MIB::applOperStatus = 2 (down) NETWORK-SERVICES-MIB::applName=<proc-name>
Unable to start call routing service.	PM giving up after too many tries to start GIS	Contact NexTone support.	SNMPv2-MIB::snmpTrapOID.0 = OID: NEXTONE-NOTIFICATION-MIB::callRoutingServiceStatus Never receive CallRoutingServiceStatus trap with applOperStatus = 1 (up) and SNMP walk of applTable shows applOperStatus of <proc> is 2 (down).
Server heartbeat	Sent at regular intervals to indicate server health	If NMS times out waiting for heartbeat, server may be down.	DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (1006) 0:00:10.06, SNMPv2-MIB::snmpTrapOID.0 = OID: NEXTONE-NOTIFICATION-MIB::heartBeat
nxCallTapDFReachabilityStatusNotify	The DF is unreachable or becomes reachable after being unreachable.	Check if the DF server name is resolvable by the iServer. Check if the DF is reachable from the iServer.	SNMPv2-MIB::snmpTrapOID.0 = OID: NX-CALLTAP-MIB:: nxCallTapDFReachabilityStatusNotify NX_CALLTAP-MIB::nxCallTapDFName=ss8CaleaServer NX_CALLTAP-MIB::nxCallTapDFPort=20120 NX_CALLTAP-MIB::nxCallTapDFReachabilityStatus = 1(reachable) NX_CALLTAP-MIB::nxCallTapDFConnectionFailureReason= 0(unknown)
nxCallTapDFConnectionStatusNotify	Connection status change	None. Message is purely informational.	SNMPv2-MIB::snmpTrapOID.0 = OID: NX-CALLTAP-MIB:: nxCallTapDFConnectionStatusNotify NX_CALLTAP-MIB::nxCallTapDFName=ss8CaleaServer NX_CALLTAP-MIB::nxCallTapDFPort=20120 NX_CALLTAP-MIB:: nxCallTapDFConnectionStatus = 1(connected) NX_CALLTAP-MIB::nxCallTapDFConnectionFailureReason= 0(unknown)
nxCallTapAFWarrantLimitStatusNotify	The AF reaches its warrant limit or recedes from its warrant limit by 10% after reaching it.	If the warrant limit is reached, new warrants cannot be added. A higher license limit should avoid this issue.	SNMPv2-MIB::snmpTrapOID.0 = OID: NX-CALLTAP-MIB:: nxCallTapAFWarrantLimitStatusNotify NX_CALLTAP-MIB::nxCallTapDFName=ss8CaleaServer NX_CALLTAP-MIB::nxCallTapDFPort=20120 NX_CALLTAP-MIB:: nxCallTapAFWarrantLimitRemaining = 10
nxCallTapAFInterceptLimitStatusNotify	The AF reaches its warrant limit or exceeds from the limit by 10% after reaching it.	When the intercept limit is reached, more calls cannot be intercepted. A higher license limit should avoid this issue.	SNMPv2-MIB::snmpTrapOID.0 = OID: NX-CALLTAP-MIB:: nxCallTapAFInterceptLimitStatusNotify NX_CALLTAP-MIB::nxCallTapDFName=ss8CaleaServer NX_CALLTAP-MIB::nxCallTapDFPort=20120 NX_CALLTAP-MIB:: nxCallTapAFInterceptLimitRemaining = 10

**Table 9. NexTone Application Traps**

Event Description	Cause	Recommended Response	SNMP Trap Objects and Example Values
sensorAlarmMesg	BMC sensor alarm triggered by hardware operational status events	Check trap for probable cause of hardware failure or abnormal status and reboot or initiate manufacturer repair.  Traps also give chassis status information, such as temperature and management subsystem health.	DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (5951462) 16:31:54.62 SNMPv2-MIB::snmpTrapOID.0 = OID: NEXTONE-NOTIFICATION-MIB::sensorAlarm NEXTONE-NOTIFICATION-MIB::sensorAlarmMesg = STRING: "0004 03/08/07 00:15:02 BMC 10 SEL Disabled 09 Log Cleared"
nexToneDOSSourceLevelThresholdAlarm	DOS attack is occurring at source level, manifested by source IPs whose signaling load is dropped at source-level rate check.	Check source logs for attack details and follow standard network attack protocol.	TBD [I asked the developer to generate the trap for inclusion here]
nexToneDOSSubnetLevelThresholdAlarm	DOS attack is occurring at subnet level, manifested by subnet IPs whose signaling load is dropped at subnet-level rate check.	Check subnet logs for attack details and follow standard network attack protocol.	TBD [I asked the developer to generate the trap for inclusion here]
nexToneDOSAtRealmLevelThresholdAlarm	DOS attack is occurring at realm level. A realm is under attack if it receives IP packets at a rate greater than the configured, allowable rate for the realm.	Check MSX logs for attack details and follow standard network attack protocol.	TBD [I asked the developer to generate the trap for inclusion here]
nexToneDOSAtSystemLevelThresholdAlarm	DOS attack is occurring at system level. A source endpoint is attacking at system level if its signaling load gets dropped at our system-level rate check.	Check MSX logs for attack details and follow standard network attack protocol.	TBD [I asked the developer to generate the trap for inclusion here]

**Table 9. NexTone Application Traps**

Event Description	Cause	Recommended Response	SNMP Trap Objects and Example Values
nexToneLayer5DOSSourceLevelThresholdAlarm	DOS attack is occurring at source level, manifested by source IPs whose signaling load is dropped at source-level rate check.	Check source logs for attack details and follow standard network attack protocol.	TBD [I asked the developer to generate the trap for inclusion here]
nexToneLayer5DOSAtRealmLevelThresholdAlarm	DOS attack is occurring at realm level. A realm is under attack if it receives IP packets at a rate greater than the configured, allowable rate for the realm.	Check MSX logs for attack details and follow standard network attack protocol.	TBD [I asked the developer to generate the trap for inclusion here]
nexToneLayer5DOSAtSystemLevelThresholdAlarm	DOS attack is occurring at system level. A source endpoint is attacking at system level if its signaling load gets dropped at our system-level rate check	Check MSX logs for attack details and follow standard network attack protocol.	TBD [I asked the developer to generate the trap for inclusion here]

a. Note that in the event of a `gis` hang, three events are sent, in the order listed in this table, starting with `Call routing service gis not responding`.

## Trap responses

Each SNMP trap has an event that triggers it. Table 10 lists events, along with a cause for each, and a detailed recommended response to the event.



Table 10. Events, Causes and Recommended Responses

Event Description	Possible Causes	Recommended Response
Free space on a monitored partition is within the threshold range.	<ul style="list-style-type: none"> <li>Log rollover/trim functions have failed</li> <li>Excessive core dump files generated</li> <li>CDR file growth off-loaded</li> </ul>	<p>a) Check the trap output to determine which partition is over the limit.</p> <p>b) Furthermore, use <code>df -k</code> and <code>du</code> commands to find out which partition, directory or files are taking up most of the space. For example:</p> <pre>msw2:~ # df -k Filesystem      1K-blocks      Used Available Use% Mounted on /dev/sda2        9325404    6184772    3140632   67% / tmpfs            2074260         4    2074256    1% /dev/shm /dev/sda6        1429648     32920    1396728    3% /tmp /dev/sda3        50177472    1478188    48699284    3% /var</pre> <p>c) <code>/home</code>: CDR files are written to <code>/home/nextone/cdrs/</code> by default. Please compress, move or delete old CDRs, if the <code>home</code> partition is nearing its threshold.</p> <p>d) <code>/var</code>: Core files are written to the <code>/var/core/</code> by default. Contact NexTone support for new cores. Compress, move, or delete old core files.</p> <p>e) Log files are written to <code>/var/log/</code> by default. Check the cron jobs (<code>crontab -l</code>) to make sure that the log rotation scripts are present; manually compress, move or delete old logs, if necessary.</p> <p>f) <code>/opt</code>: On Linux, <code>/usr/local/</code> is a link to <code>/opt/</code> and the NexTone applications are installed and run in this partition.</p>

**Table 10. Events, Causes and Recommended Responses**

Event Description	Possible Causes	Recommended Response
<p>Ethernet link state has changed.</p> <p><i>Note: Enabling this function may generate excessive traps in HA systems.</i></p>	<ul style="list-style-type: none"> <li>• Disconnected cable</li> <li>• HA system failover</li> <li>• Device at other end of link went down</li> <li>• Manual stop</li> </ul>	<p>a) If the trap indicates that link state has changed to <i>down</i>, check cabling and attached device physical connection:</p> <p>b) Use <code>dmesg</code> to determine which interface(s) has the problem</p> <p>c) If the link is to the interfaces between two servers of an HA cluster, check the other server to make sure <i>it</i> is up and running</p> <p>d) If the link is to a management, signaling or media interfaces:</p> <ul style="list-style-type: none"> <li>• Check the device at the other end to make sure it is up and running;</li> <li>• Check the cable</li> <li>• Check the NIC card</li> </ul>

Table 10. Events, Causes and Recommended Responses

Event Description	Possible Causes	Recommended Response																																																																																																																																																																																																																								
Memory allocated to a running process or used by the whole system has fallen into a threshold range.	Generally associated with a leak in the main call routing process, <code>gis</code> .  <i>Note: Correct configuration depends on installation; see the iServer release notes.</i>	a) Log on to the system. Use <code>ps -ef</code> command to locate the PID of the process.																																																																																																																																																																																																																								
		b) Use the <code>top</code> command to display the running processes and memory usage (the <code>SIZE</code> column)																																																																																																																																																																																																																								
		For example:																																																																																																																																																																																																																								
		<pre>msw2:~ # top top - 14:34:41 up 9 days, 21:47,  2 users,  load average: 0.10, 0.04, 0.00 Tasks: 105 total,  1 running, 104 sleeping,  0 stopped,  0 zombie Cpu(s):  0.0% us,  0.0% sy,  0.0% ni, 100.0% id,  0.0% wa,  0.0% hi,  0.0% si Mem:   4148524k total, 3394648k used,  753876k free,  134960k buffers Swap:  1429744k total,    0k used,  1429744k free,  2881192k cached</pre>																																																																																																																																																																																																																								
		<table><tr><th>PID</th><th>USER</th><th>PR</th><th>NI</th><th>VIRT</th><th>RES</th><th>SHR</th><th>S</th><th>%CPU</th><th>%MEM</th><th>TIME+</th><th>COMMAND</th></tr><tr><td>11104</td><td>root</td><td>16</td><td>0</td><td>2008</td><td>1064</td><td>808</td><td>R</td><td>0.3</td><td>0.0</td><td>0:00.01</td><td>top</td></tr><tr><td>1</td><td>root</td><td>16</td><td>0</td><td>664</td><td>240</td><td>204</td><td>S</td><td>0.0</td><td>0.0</td><td>0:01.24</td><td>init</td></tr><tr><td>2</td><td>root</td><td>RT</td><td>0</td><td>0</td><td>0</td><td>0</td><td>S</td><td>0.0</td><td>0.0</td><td>0:00.42</td><td>migration/0</td></tr><tr><td>3</td><td>root</td><td>34</td><td>19</td><td>0</td><td>0</td><td>0</td><td>S</td><td>0.0</td><td>0.0</td><td>0:00.00</td><td>ksoftirqd/0</td></tr><tr><td>4</td><td>root</td><td>RT</td><td>0</td><td>0</td><td>0</td><td>0</td><td>S</td><td>0.0</td><td>0.0</td><td>0:00.23</td><td>migration/1</td></tr><tr><td>5</td><td>root</td><td>34</td><td>19</td><td>0</td><td>0</td><td>0</td><td>S</td><td>0.0</td><td>0.0</td><td>0:00.00</td><td>ksoftirqd/1</td></tr><tr><td>6</td><td>root</td><td>RT</td><td>0</td><td>0</td><td>0</td><td>0</td><td>S</td><td>0.0</td><td>0.0</td><td>0:00.33</td><td>migration/2</td></tr><tr><td>7</td><td>root</td><td>34</td><td>19</td><td>0</td><td>0</td><td>0</td><td>S</td><td>0.0</td><td>0.0</td><td>0:00.00</td><td>ksoftirqd/2</td></tr><tr><td>8</td><td>root</td><td>RT</td><td>0</td><td>0</td><td>0</td><td>0</td><td>S</td><td>0.0</td><td>0.0</td><td>0:00.23</td><td>migration/3</td></tr><tr><td>9</td><td>root</td><td>34</td><td>19</td><td>0</td><td>0</td><td>0</td><td>S</td><td>0.0</td><td>0.0</td><td>0:00.00</td><td>ksoftirqd/3</td></tr><tr><td>10</td><td>root</td><td>10</td><td>-5</td><td>0</td><td>0</td><td>0</td><td>S</td><td>0.0</td><td>0.0</td><td>0:00.00</td><td>events/0</td></tr><tr><td>11</td><td>root</td><td>10</td><td>-5</td><td>0</td><td>0</td><td>0</td><td>S</td><td>0.0</td><td>0.0</td><td>0:00.00</td><td>events/1</td></tr><tr><td>12</td><td>root</td><td>10</td><td>-5</td><td>0</td><td>0</td><td>0</td><td>S</td><td>0.0</td><td>0.0</td><td>0:00.00</td><td>events/2</td></tr><tr><td>13</td><td>root</td><td>10</td><td>-5</td><td>0</td><td>0</td><td>0</td><td>S</td><td>0.0</td><td>0.0</td><td>0:00.00</td><td>events/3</td></tr><tr><td>14</td><td>root</td><td>10</td><td>-5</td><td>0</td><td>0</td><td>0</td><td>S</td><td>0.0</td><td>0.0</td><td>0:00.01</td><td>khelper</td></tr><tr><td>19</td><td>root</td><td>10</td><td>-5</td><td>0</td><td>0</td><td>0</td><td>S</td><td>0.0</td><td>0.0</td><td>0:00.00</td><td>kthread</td></tr><tr><td>32</td><td>root</td><td>11</td><td>-5</td><td>0</td><td>0</td><td>0</td><td>S</td><td>0.0</td><td>0.0</td><td>0:00.00</td><td>kacpid</td></tr></table>	PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND	11104	root	16	0	2008	1064	808	R	0.3	0.0	0:00.01	top	1	root	16	0	664	240	204	S	0.0	0.0	0:01.24	init	2	root	RT	0	0	0	0	S	0.0	0.0	0:00.42	migration/0	3	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/0	4	root	RT	0	0	0	0	S	0.0	0.0	0:00.23	migration/1	5	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/1	6	root	RT	0	0	0	0	S	0.0	0.0	0:00.33	migration/2	7	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/2	8	root	RT	0	0	0	0	S	0.0	0.0	0:00.23	migration/3	9	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/3	10	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	events/0	11	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	events/1	12	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	events/2	13	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	events/3	14	root	10	-5	0	0	0	S	0.0	0.0	0:00.01	khelper	19	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	kthread	32	root	11	-5	0	0	0	S	0.0	0.0	0:00.00	kacpid
		PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND																																																																																																																																																																																																													
		11104	root	16	0	2008	1064	808	R	0.3	0.0	0:00.01	top																																																																																																																																																																																																													
		1	root	16	0	664	240	204	S	0.0	0.0	0:01.24	init																																																																																																																																																																																																													
		2	root	RT	0	0	0	0	S	0.0	0.0	0:00.42	migration/0																																																																																																																																																																																																													
		3	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/0																																																																																																																																																																																																													
4	root	RT	0	0	0	0	S	0.0	0.0	0:00.23	migration/1																																																																																																																																																																																																															
5	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/1																																																																																																																																																																																																															
6	root	RT	0	0	0	0	S	0.0	0.0	0:00.33	migration/2																																																																																																																																																																																																															
7	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/2																																																																																																																																																																																																															
8	root	RT	0	0	0	0	S	0.0	0.0	0:00.23	migration/3																																																																																																																																																																																																															
9	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/3																																																																																																																																																																																																															
10	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	events/0																																																																																																																																																																																																															
11	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	events/1																																																																																																																																																																																																															
12	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	events/2																																																																																																																																																																																																															
13	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	events/3																																																																																																																																																																																																															
14	root	10	-5	0	0	0	S	0.0	0.0	0:00.01	khelper																																																																																																																																																																																																															
19	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	kthread																																																																																																																																																																																																															
32	root	11	-5	0	0	0	S	0.0	0.0	0:00.00	kacpid																																																																																																																																																																																																															
(cont.)																																																																																																																																																																																																																										

**Table 10. Events, Causes and Recommended Responses**

Event Description	Possible Causes	Recommended Response
memory allocated ... (cont'd from above).	(cont'd from above)	<p>c) Use <code>vmstat</code> to find out the swap memory and real memory available.</p> <p>For example:</p> <pre> msw2:~ # vmstat 2 procs -----memory----- ---swap-- ----io----- --system-- ----cpu----  r  b   swpd   free   buff   cache   si   so    bi    bo    in    cs us sy id wa  0  0       0 753572 134944 2881208    0    0     0     7     2     2  1  1 98  1  0  1       0 753572 134944 2881208    0    0     0    114 1009   117  1  0 98  1  0  0       0 753508 134944 2881208    0    0     0     38 1020    98  0  0 100  0  0  0       0 753556 134944 2881208    0    0     0      0 1003    98  0  0 100  0  0  0       0 753612 134944 2881208    0    0     0     22 1007   100  0  0 100  0 </pre> <p>d) Check the <code>/var/log/iserver.log</code> file for any errors.</p> <p>e) Make a backup copy of the <code>/var/log/iserver.log</code> file.</p> <p>f) Call NexTone support immediately.</p> <p>g) Stop and start the NexTone processes by issuing <code>allstop</code> and <code>allstart</code> from <code>bash</code> if necessary. Active calls may drop if you restart the server.</p>

**Table 10. Events, Causes and Recommended Responses**

Event Description	Possible Causes	Recommended Response
CPU idle percentage has fallen within a threshold range	<ul style="list-style-type: none"> <li>• High incoming call volume</li> <li>• Denial of service (DOS) attack</li> </ul>	<p>a) Check server for call volume using the <code>lstat</code> command under <code>bash</code>. For example:</p> <pre>msw2:~ # lstat /usr/local/nextone/bin ~ License Expiry Date      Thu Oct 26 19:00:00 2006  Licensed Features        H323    SIP    FCE    RADIUS  NAT    SIPT  Total VPORTS             2500 Available VPORTS         1097 Used VPORTS              1403  Total Media Routed VPORTS 2500 Available Media Routed VPORTS 1097 Used Media Routed VPORTS  1403</pre> <p>b) Check the <code>/var/log/iserver.log</code> file for any error messages and make a backup copy of the file.</p> <p>c) Use <code>top</code> to check the current level of CPU utilization.</p> <p>d) Use <code>vmstat</code> to check memory and other system status items.</p> <p>e) Use <code>sar -u</code> to check the CPU utilization history</p> <p>f) Use <code>tethereal</code> or <code>tcpdump</code> to initiate packet captures and check whether the server is under a DOS attack. Example of <code>tethereal</code>:</p> <pre>msw2# tethereal -w trace-eth0-1 -i eth0 Capturing on eth0 1212 ^C</pre> <p>g) Call NexTone support immediately.</p> <p>h) Stop and start the NexTone processes by issuing <code>allstop</code> and <code>allstart</code> from <code>bash</code> if necessary. Active calls may drop if you restart the server. This may also cause the server to lose data and info which is required to debug the issue.</p>

**Table 10. Events, Causes and Recommended Responses**

Event Description	Possible Causes	Recommended Response
SNMP agent startup	Normal startup	No response required
SNMP agent shutdown	Normal shutdown	No response required
SNMPv3 authentication failure	Unauthorized access or incorrect v3 configuration.	<p>a) If the user is authorized, 1) verify the password; 2) Check the SNMP V3 security configuration in <code>/var/net-snmp/snmpd.conf</code></p> <p>b) If the user is not authorized, use <code>tethereal</code> or another packet capture utility to locate the source, and take preventive action on your router/firewall.</p>
Call routing service is available	Normal startup	No response required
Host is in standby mode	A failover has occurred	This host is now backing up another host in an HA configuration. Please see the “failover has occurred” event for responses.
Host is in primary mode	A failover has occurred	This host is now the primary and active host in an HA configuration. Please see the “failover has occurred” event for responses.
Failover has occurred	Primary has failed in HA configuration	<p>a) See whether anyone has manually restarted the primary server.</p> <p>b) Check for a core dump on the primary server. Core files are located in <code>/var/core/</code> by default.</p> <p>c) Check for a network link failure using the <code>dmesg</code> command. Check:</p> <ul style="list-style-type: none"> <li>• the link between the two servers</li> <li>• signaling interfaces, if they are being monitored</li> <li>• management interface, if it is being monitored</li> <li>• media interface, if it is being monitored</li> </ul> <p><i>Note: Monitored interfaces are defined in the <code>servercfg</code> table.</i></p> <p>d) Make backup copies of the <code>/var/log/iserver.log</code> file and <code>/var/log/ispd.log</code> files.</p> <p>e) Contact NexTone support with the log files, cores files, if any, and configurations</p>

**Table 10. Events, Causes and Recommended Responses**

Event Description	Possible Causes	Recommended Response
No standby host was detected	<ul style="list-style-type: none"> <li>• No standby host</li> <li>• HA configuration compromised</li> </ul>	<p>Need to determine what happened to the standby:</p> <ol style="list-style-type: none"> <li>a) Check that the standby host is up and running, using the <code>allstat</code> command from within <code>bash</code>.</li> <li>b) Look for a network link failure between the two servers. The <code>dmesg</code> command should tell whether an interface link has been down.</li> <li>c) Check the setting of the <code>control-interface</code> and <code>peer-iserver</code> <code>servercfg</code> parameters to confirm that the peer IP address and interface are configured properly.</li> <li>d) Check and make a backup copy of <code>/var/log/iserver.log</code> file and <code>/var/log/ispd.log</code> file.</li> <li>e) Contact NexTone support with the log files and configurations.</li> </ol>

**Table 10. Events, Causes and Recommended Responses**

Event Description	Possible Causes	Recommended Response
Internal error - service restarted	iServer process error	<p>a) Run the <code>allstat</code> command under <code>bash</code> to make sure all NexTone processes are running.</p> <p>An example of <code>allstat (iserver all status)</code> output:</p> <pre>msw2:~ # allstat /usr/local/nextone/bin ~</pre> <p>Version Information:</p> <pre>----- NexTone iServer version-4.2t7, Thu Jun  1 10:36:51 EDT 2006 Copyright (c) 1998-2006 NexTone Communications, Inc.</pre> <p>Process Status:</p> <pre>-----   PID TTY      STAT   TIME  MAJFL   TRS   DRS   RSS %MEM COMMAND 7977 ?          S&lt;      0:02    1  5369 125722 15312  0.3 gis 7974 ?          S&lt;      0:00    0   747 27456 3352   0.0 dbsync 7958 ?          S&lt;      0:00    0   325 26358 3148   0.0 execd 9977 ?          S       1:15    0  2645 16858 3792   0.0 postmaster 6272 ?          S&lt;      0:16    1   471 15052 3252   0.0 pm 6269 ?          SL      0:01    2   179  3856 4036   0.0 aisexec</pre> <p>(cont.)</p>



**Table 10. Events, Causes and Recommended Responses**

Event Description	Possible Causes	Recommended Response
Internal error - service restarted (cont'd from above)	iServer process error (cont'd from above)	<p>Server Uptimes: -----</p> <p>Postgres Database Server 124 hours, 24 minutes, 33 seconds</p> <p>Nextone Process Manager 237 hours, 58 minutes, 45 seconds</p> <p>Nextone GIS Directory Server 46 hours, 28 minutes, 7 seconds</p> <p>Nextone Cmd Execution Server 46 hours, 28 minutes, 7 seconds</p> <p>Nextone DB Synchronization Server 46 hours, 28 minutes, 7 seconds</p> <p>Nextone Event Manager 237 hours, 58 minutes, 46 seconds</p> <p>License Information: -----</p> <p>License Expiry Date Thu Oct 26 19:00:00 2006</p> <p>Licensed Features H323 SIP FCE RADIUS NAT SIPT REGISTRAR CAC QOS_TAGQOS_POLICE</p> <p>Total VPORTS 100 Available VPORTS 100 Used VPORTS 0</p> <p>Total Media Routed VPORTS 100 Available Media Routed VPORTS 100 Used Media Routed VPORTS 0</p> <p>(cont.)</p>

**Table 10. Events, Causes and Recommended Responses**

Event Description	Possible Causes	Recommended Response
Internal error - service restarted (cont'd from above)	iServer process error (cont'd from above)	<p>b) Use the <code>allstop</code> and <code>allstart</code> commands under <code>bash</code> to restart the NexTone processes if they are not running. Please contact NexTone support before doing this if you want NexTone to diagnose what is happening. A restart may wipe out important data/information required to diagnose and fix the problem.</p> <p>c) Check <code>/var/log/iserver.log</code> and make a backup copies of this file.</p> <p>d) Contact NexTone support with the log files.</p>
Call routing service not responding	GIS is hung	<p>When this happens, the PM (process manager) automatically restarts the GIS in about 45 seconds. Usually, there is a core file associated with this event.</p> <p>a) Use the <code>allstat</code> command under <code>bash</code> to see if the server is running.</p> <p>b) Use the <code>allstop</code> and <code>allstart</code> commands under <code>bash</code> to restart the NexTone processes if they are not running. Please contact NexTone support before doing this if you want NexTone to diagnose what is happening. A restart may wipe out important data/information required to diagnose and fix the problem.</p> <p>c) Check the core dir (<code>/var/core/</code>) to see if there is a new core file.</p> <p>d) If there was a core, issue a <code>df -k</code> command to make sure the <code>/var</code> partition is not full. Move the core files to a different directory, or server, if the partition is nearing 100%.</p> <p>e) Make a backup copy of the <code>/var/log/iserver.log</code> file.</p> <p>f) Open a support ticket for this incident with the log files and access to core files if any.</p>

**Table 10. Events, Causes and Recommended Responses**

Event Description	Possible Causes	Recommended Response
Unresponsive call routing service has been restarted	GIS is hung	<p>When this happens, the PM (process manager) automatically restarts the GIS in about 45 seconds. Usually, there is a core file associated with this event.</p> <ul style="list-style-type: none"> <li>a) Use the <code>allstat</code> command under <code>bash</code> to see if the server is running.</li> <li>b) Use the <code>allstop</code> and <code>allstart</code> commands under <code>bash</code> to restart the NexTone processes if they are not running. Please contact NexTone support before doing this if you want NexTone to diagnose what is happening. A restart may wipe out important data/information required to diagnose and fix the problem.</li> <li>c) Check the core dir (<code>/var/core/</code>) to see if there is a new core file.</li> <li>d) If there was a core, issue a <code>df -k</code> command to make sure the <code>/var</code> partition is not full. Move the core files to a different directory, or server, if the partition is nearing 100%.</li> <li>e) Make a backup copy of the <code>/var/log/iserver.log</code> file.</li> <li>f) Open a support ticket for this incident with the log files and access to core files if any.</li> </ul>
Call routing service is stopping	GIS is being shutdown by PM	<p>PM automatically restarts the GIS if the PM misses 3 consecutive keepalive messages from GIS with interval of 15 sec. Usually, there is a core file associated with this.</p> <ul style="list-style-type: none"> <li>a) Use the <code>allstat</code> command under <code>bash</code> to see if the server is running.</li> <li>b) Use the <code>allstop</code> and <code>allstart</code> commands under <code>bash</code> to restart the NexTone processes if they are not running. Please contact NexTone support before doing this if you want NexTone to diagnose what is happening. A restart may wipe out important data/information required to diagnose and fix the problem.</li> <li>c) Check the core dir (<code>/var/core/</code>) to see if there is a new core file.</li> <li>d) If there was a core, issue a <code>df -k</code> command to make sure the <code>/var</code> partition is not full. Move the core files to a different directory, or server, if the partition is nearing 100%.</li> <li>e) Make a backup copy of the <code>/var/log/iserver.log</code> file.</li> <li>f) Open a support ticket for this incident with the log files and access to core files if any.</li> </ul>

**Table 10. Events, Causes and Recommended Responses**

Event Description	Possible Causes	Recommended Response
Unresponsive call routing process is being dumped	GIS is hung - PM is asking it to dump memory for debugging	<p>This indicates that the GIS is core dumping, or has already core dumped.</p> <ol style="list-style-type: none"> <li>Use the <code>allstat</code> command under <code>bash</code> to see if the server is running.</li> <li>Use the <code>allstop</code> and <code>allstart</code> commands under <code>bash</code> to restart the NexTone processes if they are not running. Please contact NexTone support before doing this if you want NexTone to diagnose what is happening. A restart may wipe out important data/information required to diagnose and fix the problem.</li> <li>Check the core dir (<code>/var/core/</code>) to see if there is a new core file.</li> <li>If there was a core, issue a <code>df -k</code> command to make sure the <code>/var</code> partition is not full. Move the core files to a different directory, or server, if the partition is nearing 100%.</li> <li>Make a backup copy of the <code>/var/log/iserver.log</code> file.</li> <li>Open a support ticket for this incident with the log files and access to core files if any.</li> </ol>
Call routing service is being stopped.	PM is abruptly stopping the GIS.	<p>This indicates that the PM is stopping the GIS.</p> <ol style="list-style-type: none"> <li>Use the <code>allstat</code> command under <code>bash</code> to see if the server is running.</li> <li>Use the <code>allstop</code> and <code>allstart</code> commands under <code>bash</code> to restart the NexTone processes if they are not running. Please contact NexTone support before doing this if you want NexTone to diagnose what is happening. A restart may wipe out important data/information required to diagnose and fix the problem.</li> <li>Check the core dir (<code>/var/core/</code>) to see if there is a new core file.</li> <li>If there was a core, issue a <code>df -k</code> command to make sure the <code>/var</code> partition is not full. Move the core files to a different directory, or server, if the partition is nearing 100%.</li> <li>Make a backup copy of the <code>/var/log/iserver.log</code> file.</li> <li>Open a support ticket for this incident with the log files and access to core files if any.</li> </ol>

**Table 10. Events, Causes and Recommended Responses**

Event Description	Possible Causes	Recommended Response
Unable to start call routing service.	PM giving up after too many tries to start GIS	<p>Check the following system configuration settings:</p> <ul style="list-style-type: none"> <li>a) iServer shared memory settings in <code>/etc/sysctl.conf</code></li> <li>b) The <code>h323-maxcalls</code> attribute setting in <code>servercfg</code></li> <li>c) The <code>h323-maxcalls-sgk</code> attribute setting in <code>servercfg</code></li> <li>d) Make sure the number of routes and bindings combined on the system doesn't exceed the maximum allowed number. Use <code>cli db info</code> to find out how many routes and binding are currently configured.</li> <li>e) Check the <code>/var/log/iserver.log</code> for any error messages and make a backup of this file.</li> <li>f) Contact NexTone support with the log file and configuration files</li> </ul>
Server heartbeat	Sent at regular intervals to indicate server health	<p>If your NMS times out waiting for a heartbeat, the server may be down. Check the following:</p> <ul style="list-style-type: none"> <li>a) Verify that the server is up and running.</li> <li>b) Verify that the management interface is up by pinging the interface's IP address</li> <li>c) Verify that the NexTone SNMP agent is running, with the following command: <code>/etc/init.d/snmpd status</code></li> <li>d) If the agent is not running, please restart the agent with the following command: <code>/etc/init.d/snmpd start</code></li> <li>e) Contact NexTone support if the above steps don't resolve the issue.</li> </ul>
BMC sensor alarm	Sent at regular intervals to indicate sensor chassis status and health	<ul style="list-style-type: none"> <li>a) Read equipment status in RSM.</li> <li>b) For chassis alarms like overheating and system board failures, send unit to manufacturer for repair or replacement. For network alarms check network connectivity and security event details and resecure network.</li> </ul>
IServer DOS attacks	Sent at regular intervals to inform when a DOS attack has occurred.	<ul style="list-style-type: none"> <li>a) Check logs of IP address reporting the Denial of Service (DOS) attack and perform standard DOS attack protocol.</li> </ul>

## Configuring SNMP-related options for rate limiting

When the MSX rate limit monitoring process receives log information that either IP or session layer rate limits have been exceeded, it generates an SNMP trap. If a rate limit is exceeded because of a DoS attack, the trap information can be helpful in addressing the problem. For example, if you identify the source of the attack, you can add that source to your blacklist to prevent it from accessing your system (see *Blacklisting sources to prevent system access* on page 345 for more information on this process). MSX provides commands you can use to configure MSX interaction with SNMP for rate limiting. You can use these commands to control the number of traps and how many entries appear in the TopN tables.

### Configuring the SNMP trap threshold

By default, MSX generates only one trap, even if the rate limits for an object continue to be exceeded as might occur during a DoS attack. You can configure a time interval that controls how frequently additional traps are generated for the same object. You can define an interval at different levels within the system using the `cli thresholdcross edit` command as follows:

```
cli thresholdcross edit <ep/subnet/realm/system/session> timeout <value>
```

where:

you can choose the level for the time interval: `ep` (endpoint), `subnet`, `realm`, `system`, or `session`.

`timeout <value>` is the minimum number of seconds you want to occur between SNMP traps referring to the same object.

The default value is 0 for each level which means that MSX generates only one trap.

For example, to change the interval to 10 seconds between traps for realms that are experiencing a sustained series of rate limit violations, use the following:

```
cli thresholdcross edit realm 10
```

To list the current settings for each level, use:

```
cli thresholdcross list
```

### Configuring TopN SNMP tables

The amount of SNMP trap data can grow to be quite large and hard to use effectively. You can configure how many entries are taken from the base SNMP tables to create corresponding “topN” tables. These topN tables contain the most frequent attackers from different types of sources, and you configure how

many entries (topN) to include in the table. To configure the tables use the `cli ratelimittopn edit` command as follows:

```
cli ratelimittopn edit <ip/session> <type> <value>
```

where:

`specifying` either `ip` or `session` indicates at which layer the rate limiting errors recorded in the table occurred.

`type` is the type of DoS attack and corresponds to a specific TopN SNMP table that contains information specifically on that type of attack. Specify a `type` to configure the size of the corresponding TopN file. The valid options you can use with `ip` are:

- `source` - information on endpoints exceeding their limits
- `subnet` - information on subnets exceeding their limits
- `sourceInSubnet` - information on endpoints exceeding subnet limits
- `realm` - information on which realms are having their limits exceeded
- `sourceInRealm` - information on endpoints exceeding realm limits
- `sourceInSystem` - information on endpoints exceeding system limits

and the valid options you can use with `session` are:

- `source` - information on endpoints exceeding their limits
- `realm` - information on which realms are having their limits exceeded
- `sourceInRealm` - information on endpoints exceeding realm limits
- `sourceInSystem` - information on endpoints exceeding system limits

`<value>` is the number of entries to include in the topN table.

For example, to include 10 entries in the topN SNMP table associated with realms that have exceeded their rate limits in the `ip` layer, use the following:

```
cli ratelimittopn edit ip realm 10
```

## References

IETF Request for Comments (RFC) references may be found at <http://ietf.org/rfc.html> or at <http://www.faqs.org/rfcs>.

<http://www.net-snmp.org> - General information about the Net-SNMP package

**RFC 1157** – *Simple Network Management Protocol*

**RFC 1213** – *Management Information Base for Network Management of TCP/IP-based internets:MIB-II*

**RFC 2981** – *Event MIB*

**RFC 3413** – *Simple Network Management Protocol (SNMP) Applications*

**RFC 3414** – *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMP)*

**RFC 3415** – *View-based Access Control Model (VACM) for the Simple Network Management Protocol*



## MSX Redundancy

### Introduction

The NexTone MSX supports 1+1 redundancy at two levels: machine and data. These are described below.

#### 1+1 redundancy

A 1+1 redundancy configuration provides switchover (sometimes called *failover*) redundancy between two MSX machines specifically paired for this purpose. In this configuration, at a given point in time only one of the machines is actively processing calls. The other is a pure standby; that is, in its standby role, it is not actively processing calls. See Figure 15.

Figure 15. 1+1 Redundancy

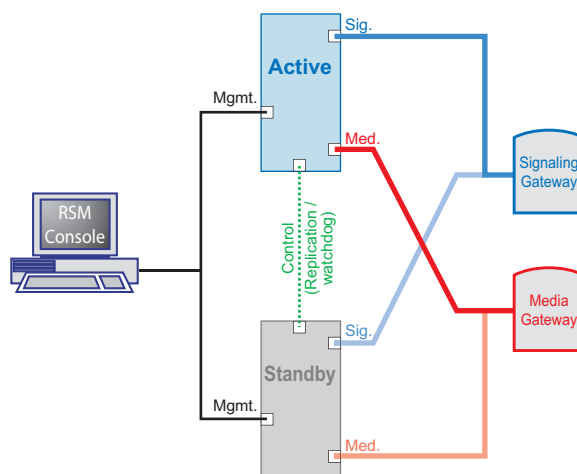


Figure 15. depicts the relationships between the major components in a 1+1 redundant system. The communication entities shown are described in “Network-level redundancy architecture”, below.

#### Machine redundancy

MSX supports “switchover” redundancy—a facility whereby one dedicated *standby* machine can take over processing responsibilities for one *active*

machine that goes out of service. This is known as “1+1 active/standby sparing”; that is, each active processor has one permanently mated standby, which only comes on-line when the active processor either fails or is administratively taken off-line. When that happens, the former *standby* becomes the new *active* until another switchover occurs, to put them back the way they were. Only one of the two machines (the *active*) is processing calls at any time. The two machines together are generally known as a *cluster*, and in the context of call processing are called a *service node*.

Note that a machine switchover has a temporary adverse effect on call processing performance, and should not be purposely invoked unless necessary.

### **Data redundancy**

In addition to covering for a failed machine as described above, each machine hosts its own complete copy of the MSX database. While the status of the machines in a cluster are named *active* and *standby*, to avoid confusion, the database copies are referred to as *master* and *slave*.

The master database is the one which at any moment is being used by the active processor as its source of data for routes, calling plans, configured endpoints, etc. It is also the copy of the database that receives updates first, and from which they are subsequently (in near real-time) propagated to the slave copy.

To ensure synchronization between the two MSX machines, data in the master database is dynamically replicated to the slave in the background at all times during normal operation. The databases are independent of machine status (active or standby); that is, as long as a machine is running, it can host either the master or the slave database, whether that machine is the active or the standby. Upon switchover, the standby machine takes over for the active, using the data in the master database (whichever machine it's on).

If the former active machine had the master database, and that machine has now failed in such a way that the database on it is no longer running (or cannot be reached by the standby machine that became the active), the new active machine also declares that *its* copy of the database is the master. When the former active machine returns to service, its database becomes the slave, and changes to the master since it went down are propagated back to it.

## **Dynamic endpoint registration**

On a redundant system, when an endpoint dynamically registers with the MSX, it does so through the RSA for the realm in which that endpoint is located. This

endpoint state data is also actively replicated from the active processor to the standby.

## Call migration

The migration of calls (both in-progress and those being set up) is described in *Stateful Call Migration (SCM)* on page 164. Note that this feature only applies to pure SIP (i.e., SIP-to-SIP) calls, not IWF or pure H.323.

## Network-level redundancy architecture

From a service perspective, the MSX database is totally hidden from the endpoints requesting service. Thus, even when an MSX fails and is replaced by its dedicated standby system, the cluster still appears as one logical device (a service node) to the endpoints. The endpoints request service from an MSX node, based on the information in the service node's database, using the node's realm signaling and media addresses (RSA and RMA). Upon primary MSX processor failure, the standby MSX switches in for the failed primary, and begins servicing all the same realm addresses. It uses the database data that was replicated onto it over the control interface while the primary was still running. Call setup and tear-down services to registered endpoints are not lost, since the realm's signaling and media addresses are still being serviced. In this way, the network is virtually redundant, since the addresses on it are always serviced, whether from the primary MSX machine or its standby.

Note that the control LAN connection can consist of a single Ethernet cable between the control interfaces on the primary and standby machines. No router or switch is actually required, and for maximum reliability, this method of connection is preferred.

### ***How to recover from a loss of control interface connectivity***

If communication between the control interfaces of a high-availability pair of MSXs is lost, whether by the cable being pulled out or some other means, use the steps here to restore service.

1. Log on as `root` to the MSX you wish to become the standby machine.
2. Stop the MSX processes on that machine by entering:  

```
iserver all stop
```
3. Reconnect the control interface cable between the MSXs.
4. Restart the MSX processes by entering:

```
iserver all stop
```

Using this procedure, when the server you are logged in to comes back on line, it will detect that the *other* MSX is already running as the active machine, and will make itself the standby.

## Database replication details

While a paired active and standby MSX are running, the service, and server and endpoint configuration databases are kept synchronized by sending database updates over the control interface from the primary to the standby. This includes both static and dynamic information. Server configuration changes can be made using either the `nxconfig.pl` utility, or RSM Lite or RSM Console.

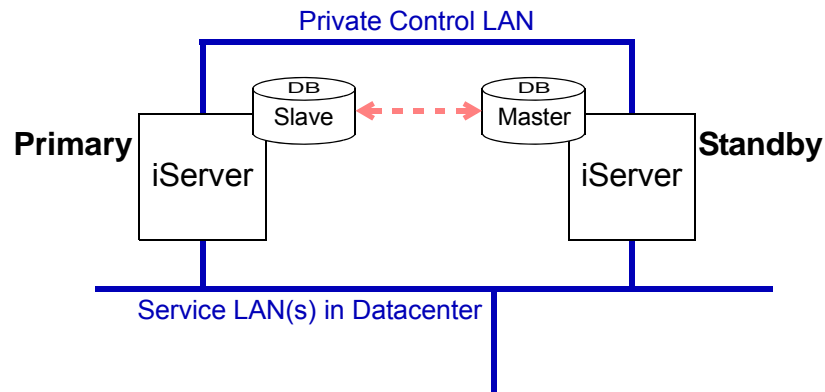
Dynamic information updates between the primary and standby systems are fairly lightweight and require little network and processing overhead. In addition, all of these changes are propagated over a private LAN (the *control* LAN) that does not carry any service traffic.

Static or provisioning updates are controlled by an administrator. These updates can vary. The databases on both the primary and the standby contain the complete information for the MSX node. For example, whenever the administrator changes the active database, the change is automatically propagated to the standby database. Since no assumptions have been made of primary and standby, the database being updated by the administrator could reside either on the primary MSX or the standby MSX. Database updates are this way made independent of the system that currently is the primary.

## Implementing redundancy

The 1+1 redundancy implementation for MSXs is depicted in Figure 16. , which shows one possible state of the cluster with respect to primary vs. standby processor, and master vs. slave database.

Figure 16. Implementing 1+1 Redundancy



While each MSX machine in the service node has a complete copy of the MSX database, at a given point in time one copy is the *master*, and the other, the *slave*. Most of the time the user and administrator do not need to know which is which. CLI and RSM Console operations are automatically made to the master, and propagated to the slave when the command is run. Remember that there is no direct correlation between the master and slave databases, and the active or standby server machines. The master can reside on either the active machine or the standby, and vice-versa.

However, some operations (such as some of those described in the chapter, “*The MSX Database*”) require that MSX processes be *quiesced* (i.e., stopped) before performing them. In that case, it makes sense to do the work on the standby machine, while the master database is on the standby server, so that service remains uninterrupted. This allows the administrator to update the database on the standby system, which is not servicing calls. The updates are then propagated to the primary system.

**Caution:** *While the MSX processes are quiescent, the standby machine cannot take over if the active machine fails.*

## ***Determining and changing status***

Ordinarily, when the machines in a redundant cluster are brought up, the machine that comes up first becomes the *active* processor, and has the *master* database, just as if it was the only processor in a single-machine setup. When this is not the desired configuration, it can be changed. The subsections below tell how.

### **Determining which machine has the master database**

There are two ways to determine which machine is currently hosting the master database: one from RSM Console, and the other from the command line.

#### ***RSM Console***

To see it in **RSM Console**, simply right-click the cluster's icon in the map window, and choose **Cluster Info...** In the **Cluster Information** window that pops up, the entry under **Database Cluster** that has the yellow database icon next to it is the current master.

#### ***CLI***

To determine it from the **command line**, log onto the machine as root, and enter the command:

```
cli rsd list
```

The command returns two lines, of the format:

```
IP: ip address Status: status
```

The machine on which the command is run appears first in the list, followed by its mated standby. Values for status are Master or Slave.

### **Changing master database hosting**

Once you have determined which machine is hosting the master database, if you need to change it, log onto the host that currently has the master, then enter the following command:

```
iserver rsd stop; sleep 15; iserver rsd start
```

This is in fact three commands, that when run force a failover. When the slave database machine senses that it has lost connection to the master (during the 15-second sleep), it proclaims itself the new master, and the switch-over is accomplished.

### **Determining the active processor**

To determine which iServer is currently processing calls (i.e., the active iServer), right-click the cluster's icon on the RSM Console map window, and choose **Cluster Info...** In the **Cluster Information** window that pops up, the

entry under **Network Cluster** that has the little network icon next to it is the current active processor.

### **Changing the active iServer**

At this time, the only ways to force a failover from the active iServer to the standby are not “graceful.” That is, though a failover will result, the methods are likely to produce undesirable side-effects, the least of which is service outages (approximately 45 seconds) while the standby machine switches in as the active. Therefore, this work should be appropriately scheduled to minimize service impact.

The method deemed best for forcing a failover is to shut down all processes that provide MSX services (call signaling, routing, etc.). In order to do this, log onto the active processor as `root`, and enter:

```
iserver all stop
```

This initiates the switchover, which can take most of a minute or more to actually complete. To confirm the system’s up, log onto it as `root`, and enter:

```
iserver gis status
```

Look in the output for something like the following:

Process Status:

```
-----
F S  UID  PID  PPID  C PRI NI   ADDR      SZ    WCHAN TTY      TIME CMD
8 S   0   425    1   0  40 10 e7eba078 409112 e74b9242 ?        0:11 gis
```

...where `gis` is listed (all the way to the right), and you *don't* get a message that says `iServer not running`.

### ***Implementation notes and definitions:***

This section discusses some additional details concerning implementation.

- As mentioned above, stateful call migration is implemented only for SIP calls. Call processing for H.323 involves setting up TCP connections through the MSX. In most TCP/IP stacks, TCP connections cannot be migrated to another host. Hence, during a transition, all TCP-based calls (i.e., H.323 calls) being processed by the primary are lost.
- MSX peers - The MSXs in a redundant pair are considered peers to each other. There is no notion of peer discovery. Each MSX’s *servercfg* table contains the name of its peer MSX.
- Service LAN - The LAN carrying signaling and/or media call traffic. Gateways and endpoints must be able to connect to this LAN.

- **Control LAN - Redundancy** in the MSX is accomplished by sending RPC traffic on a *control* LAN between the MSX peers. Heartbeats are sent by peers every 2 seconds. These interfaces are most often connected via a single network cable for maximum reliability.
- **Primary MSX behavior** - The primary MSX performs the following actions:
  - Responds to heartbeats from the standby.
  - Initiates the switch-over to the standby peer whenever it detects a network interface failure, or when the network link goes down.
  - Checks the status of the *gis* process every 2 seconds. If the *gis* process is not present or does not respond, then it initiates the switch to the standby peer.
  - Initiates the switch in of the standby peer via a private **ALARM** command sent to the peer *ispd* process on the standby.
- **Standby MSX behavior** - The standby MSX sends heartbeats to its active peer. A heartbeat is sent every 2 seconds, and if five heartbeat responses are missed, the standby initiates a failover.
- **Database replication** - Database replication is accomplished using RPCs over the control LAN.

Two types of data updates are possible: bulk and incremental. Incremental updates are propagated across MSX systems via command line interface (CLI) commands (See “*The MSX Database*” on page 106 and *The Command Line Interface* on page B-1 for details). Bulk updates use the *rsync* protocol. Failure of an incremental update automatically triggers a bulk update. To avoid conflicting updates, all real-time updates must be made to the master.

## Configuring a redundant MSX

When setting up 1+1 redundancy, the CPU speed, amount of memory available, and disk space of both MSX host machines should be the same.

Redundant MSX systems are configured in this order:

1. Configure the servers’ system time attributes
2. Configure the network interfaces
3. Configure database replication



### **Configuring the servers' system time attributes**

For redundancy to work correctly, both servers must have the same system time and time zone settings. NTP is useful for assuring system time synchronization. For more information on NTP, go to [www.ntp.org](http://www.ntp.org). Several other MSX functions also require the same kind of tight system clock synchronization, such as *Stateful Call Migration (SCM)* on page 164, *H.235 security authentication* on page 279 and *Vocaltec support considerations* on page 285.

### **Configuring replication network interfaces**

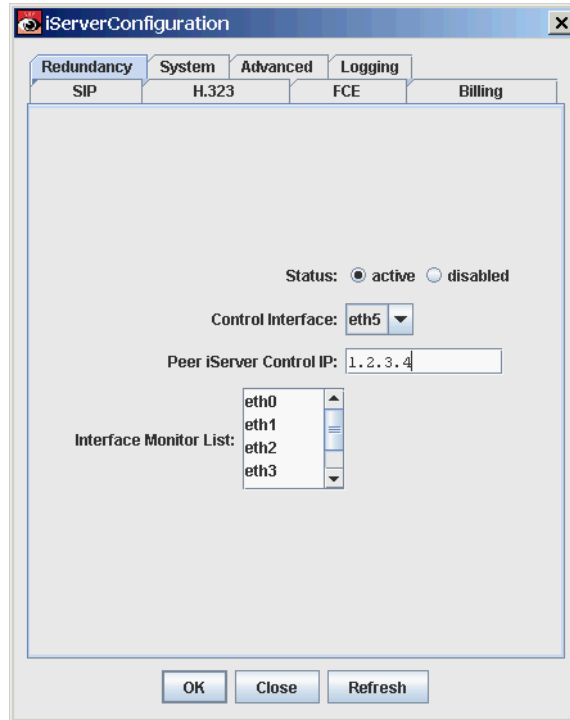
In a redundant configuration, each MSX host must have at least two network interfaces. One interface connects to the *service LAN* and the other connects to the *control LAN*. The signaling and media traffic destined to the MSX is carried over the service LAN, and must be reachable by all the endpoints. The control LAN may be privately addressed. For best reliability, the peer systems should be directly connected with an Ethernet cable. Both broadcast and multicast must be enabled on both the service and control LANs.

To specify the interface to use, see the `control_interface` entry in *Configuring the control interface with `nxconfig.pl`* on page 162.

### **Examining the control interface from RSM console**

RSM Console provides a facility to view redundancy control settings. Note that these parameters are set with the `nxconfig.pl` command (see *Configuring the control interface with `nxconfig.pl`* on page 162). To view the settings:

1. From RSM Console's main map window, right-click on the server's icon, choose *Configure... -> Redundancy -> Network*. Figure 17. shows this panel.

**Figure 17. RSM Console's Control Interface Settings**

2. Ensure the active radio button is selected if the system is redundant and you want it to run in redundant mode.
3. The **Control Interface** box shows the name of the physical interface used as the control interface.
4. The **Peer iServer Control IP** box shows the IP address of the *peer* machine's control interface.
5. The **Interface Monitor List** shows the interfaces that the iServer monitors for failure. Failure on a monitored interface triggers a failover to the peer machine.
6. Click **OK** or **Close** when finished.

### **Configuring the control interface with *nxconfig.pl***

Four MSX parameters relate to peering in a redundant MSX pair. These are described in Table 11, and can only be set with the *nxconfig.pl* utility.

**Table 11. peering\_config Block Fields**

Parameter Name	Description	Comments
server_type	An attribute that allows a machine to be taken "off line," that is, made unavailable, for any reason.	Valid values: "active" or "disabled" If set to active, stateful call migration is also enabled.
control_interface	The name of the control interface used to communicate with a peer MSX.	For example: "eth1"
interface_monitor_list	A space-separated list of interface names, in quotes.	For example: "eth0" "eth1" "eth2"
peer_iserver	The IP address of this machine's peered standby machine	Enclose in double quotation marks

To set these parameters, log into the iServer and enter:

```
nxconfig.pl -e parameter -v value
```

where parameter is a parameter from Table 11 and value is the setting you are giving to the parameter.

**Note:** *At the least, the active processor must have a server\_type of active. If the standby machine is set to disabled, the active machine will still run, but the standby will not take over if the active fails. If both are inadvertently set to disabled, the service node will not come up.*

### Configuring database replication

Database replication occurs on two levels: *incremental* and *bulk*. Incremental updates are performed over a multicast protocol between the peers. Bulk updates are performed as needed and require the use of rsync, which is installed along with the iServer Admin Package.

1. All database updates, whether incremental or bulk, are only handled over the LAN interface specified in the table. Ensure that this interface is the same as for the Control LAN.
2. The rsync-test script is provided in /usr/local/nextone/etc. Run this script to verify that the rsync package has been properly installed.
3. A multicast address and port must be configured on the Control LAN of each MSX machine. They must be the same for each MSX machine. The multicast address and port are configured using sconfig.

4. The application automatically adds a multicast route on the Control LAN interface. The multicast route should not be changed or removed, and care must be taken to ensure that there are no conflicts with the route.
5. A priority field that sets the priority of the MSX to dynamically determine who is the master of the cluster of MSXs. All *bulk* updates emanate from the master. If the priorities of the MSX systems are equal, the master election algorithm selects the MSX with the longest uptime.

## Stateful Call Migration (SCM)

This SIP feature prevents dropping of active calls during a switchover.

Without SCM, when a redundancy switchover takes place, the switch that's shutting down drops in-progress calls. The CDRs for such dropped calls are marked with "shutdown" (error code value=1016) in the *call-error* CDR field (#14). With Stateful Call Migration (SCM), call state information is retained during a switchover, and active calls are maintained rather than dropped.

Each MSX in a redundant pair maintains its own copy of the call state information. This information is replicated to the standby MSX during normal MSX operation. This data on the two machines is designated as *Master* on one, and *Slave* on the other; changes are made to the master, and replicated to the slave, with minimal latency, using RPC (remote procedure calls).

### ***Additional replicated data***

SCM involves replicating some data not replicated before SCM was supported. These items include:

- CAC (Call Admission Control) data  
The current-version MSX provides for CAC across multiple endpoint/ports. Aggregate totals for calls in a CAC group are replicated to the slave policy database so that the standby machine knows how to limit calls to that endpoint group.
- Session Timers and Call Duration timers  
One of these timers being started, but not carried forward to the standby server during call setup or continuance, would produce unexpected results, so these timers are replicated.

### **Implementation highlights**

- Replication of call state data between master and slave databases follows the well-known add/change/delete paradigm. That is, the system is either replicating a new call setup (known as a New State), updating the status of an existing call (*State Update*), or removing from the database a formerly active call (*Delete State*).
- Note that the *primary* call-state database actually resides on the *standby* server, not the primary server. This is so that database update processing has minimal impact on call processing performance.
- During normal operation, to minimize impact on call processing performance, call state-data migration takes place *after the call is connected*. Note that if a call is set up, then torn back down before replication takes place (if the standby server is temporarily down, for example), no call state data are replicated to the standby.
- Call leg replication begins when primary server detects standby server's heartbeats.
- SCM is backward and forward compatible across differing MSX releases.
- Calls that are not replicated to the standby server are still correctly disconnected, and CDRs correctly created.
- SCM is disabled by default.

### **Implementation requirements**

For SCM to work, the requirements in this section apply.

#### **NTP**

To ensure proper operation of timers and other features that rely on precise timing (such as CDRs), NTP (network time protocol) is required for both MSXs in the redundancy pair, and of course, both servers must use the same NTP master, preferably one outside of either server, in case of a failure. Using NTP ensures that the CDR on the standby machine is accurate and each call has one, complete, authoritative CDR.

### **Checking SCM status**

A new `cli` command is provided to obtain status information for call states under SCM's control. Its format is:

```
cli scm
```

The output from the command is:

Signaling State	active
SCM Total States	0
SCM Pending States	0
SCM States successfully sent	0
SCM States failed	0

**Note:** *Each call contributes two states (one for each leg) to the above counts.*

**signaling state** shows whether the ispd daemon is running (i.e., whether the machine is the active server or the standby).

**total states** indicates the total number of call *legs* active on the MSX on which the command is run.

**pending states** indicates call legs that are set up on the primary MSX, but which haven't yet been replicated to the standby MSX.

**states successfully sent** is the cumulative total of call legs sent from the primary MSX that were successfully replicated on the standby since the last switchover, irrespective of whether the call is still in progress.

**states failed** indicates the number of call legs that were not successfully replicated to the standby MSX since the last switchover. A non-zero value here may indicate the standby server is down. *(This feature is presently disabled, so it will always read zero.)*

### Caveats

- Currently, **SCM only applies to pure SIP calls**. This is because SIP only uses UDP; not TCP, which H.323 requires. UDP, being the simpler protocol, has been implemented. The SIP leg of an IWF call is replicated to the standby, but during switchover, that leg is closed.
- While **CAC data** is not dynamically maintained with complete accuracy (owing to latency, not errors as such), it will be correct at the point when the standby server becomes the active server.
- The MSX no longer pings the routers involved with redundancy; the customer assumes responsibility for ensuring their availability.
- Regarding redundancy:
  - The signaling interfaces are defined in the `mdevices.xml` file and the realm configuration (via the Realms utility).
  - Database ownership is independent of the server's status as active or standby. If either server is down, updates to the policy database can still be made on the running server.
  - The VIP module feeds events into the Call Migration Module.

## IPSec Support

### About IPSec

- This chapter describes the use of ESP protocol, IKE key exchange, and pre-shared keys
- The iServer uses the `racoon`<sup>1</sup> IPSec client. Devices running other clients are supported *if* they perform IKE-based IPSec operations.

**Note:** *The procedure described in this chapter for implementing IPSec is the only recommended procedure. If you deviate from the configuration outlined in this chapter, responses to incoming packets will not be sent. Follow the implementation instructions as presented to ensure proper operation.*

### How to implement IPSec

The steps in the rest of this section are based on the example of an iServer and proxy server being configured to use IPSec to communicate with each other. Note that in this example, both devices are using the `racoon` client. If your non-iServer device is using another client, you must perform the procedure appropriate for that client, which is not given here, but the principles presented apply to any IPSec client.

For the example, assume:

- the interface on the iServer is 192.168.1.3
- the interface on the proxy is 192.168.1.2
- the pre-shared key between the iServer and the proxy is `foobar`

Your installation will substitute the IP addresses for your own equipment where the above addresses appear.

**Note:** *Before beginning the actual procedure on your machines, be sure that you have properly configured interfaces and routes on all involved systems, and that they can reach each other. Ping the interfaces from both ends, to confirm that all interfaces have connectivity—then proceed.*

---

1. Yes, that is *racoon*, spelled with only one “c”.

The procedure requires editing text files on the iServer, using an editor you are familiar with. (Specific key sequences are not given, since they depend on the editor you choose to use.) If you are not comfortable with doing this, please either obtain the assistance of someone you know who is, or contact NexTone support for help.

### ***Procedure for configuring IPSec on the iServer***

1. Edit the file `/etc/racoon/psk.txt`, and add the pre-shared key, by adding this line to the end of the file:

```
192.168.1.2 foobar
```

Save the file and quit the editor.

2. Edit the file `/etc/racoon/setkey.conf`, adding this key configuration information to the end of the file (be sure to include the semicolons at the end of each line):

```
spdadd 192.168.1.3 192.168.1.2 any -P out ipsec esp/transport//require;  
spdadd 192.168.1.2 192.168.1.3 any -P in ipsec esp/transport//require;
```

Save the file and quit the editor.

3. Edit the file `/etc/racoon/remote.conf`, adding the endpoint configuration information shown below to the end of the file. If the file doesn't exist, create it. We recommend you copy this and paste the lines into your file, then change the IP addresses accordingly. Save your changes and quit the editor.



4.

```

remote 192.168.1.2 {
    exchange_mode main;
    lifetime time 24 hour;
    dpd_delay 10; //Set the dead peer detection timer
    proposal {
        encryption_algorithm 3des;
        hash_algorithm md5;
        authentication_method pre_shared_key;
        dh_group 2;
    }
}

sainfo address 191.168.1.3 any address 192.168.1.2 any {
    lifetime time 1 hour;
    encryption_algorithm 3des;
    authentication_algorithm hmac_md5;
    compression_algorithm deflate;
}

sainfo address 191.168.1.2 any address 192.168.1.3 any {
    lifetime time 1 hour;
    encryption_algorithm 3des;
    authentication_algorithm hmac_md5;
    compression_algorithm deflate;
}

```

5. For the endpoint configuration defined in step 3 to take effect, the file `/etc/racoon/remote.conf` must be included in `/etc/racoon/racoon.conf`. Open `/etc/racoon/racoon.conf` for editing, and search for `include remote.conf`. If it is commented out (that is, if the line starts with a `#` sign), delete the `#`.
6. While you have `/etc/racoon/racoon.conf` open, search the entire file for the string `aes` and remove it from any line that it appears in. Don't delete any entire line; just delete `aes` from it. (The default racoon configuration file includes AES support, but the iServer's Linux kernel does not support the AES algorithm.)  
Save any changes you made, then quit the editor.
7. You have now completed configuration activities for IPSec support on the iServer. To activate the services with the new configuration, start racoon by entering:

```
/etc/init.d/racoon start
```

8. You may now log off of the iServer.

### ***Procedure for configuring IPSec on the proxy***

The procedure for configuring the proxy's support is nearly identical to that for the iServer, except that the IP addresses for the respective devices are adjusted to allow for the proxy being on the other end of the connection.

The procedure, then, is:

1. Edit the file `/etc/racoon/psk.txt`, and add the pre-shared key, by adding this line to the end of the file:

```
192.168.1.3 foobar
```

Save the file and quit the editor.

2. Edit the file `/etc/racoon/setkey.conf`, adding this key configuration information to the end of the file (be sure to include the semicolons at the end of each line):

```
spdadd 192.168.1.2 192.168.1.3 any -P out ipsec esp/transport//require;  
spdadd 192.168.1.3 192.168.1.2 any -P in ipsec esp/transport//require;
```

Save the file and quit the editor.

3. Edit the file `/etc/racoon/remote.conf`, adding the endpoint configuration information shown below to the end of the file. (If the file doesn't exist, create it. We recommend you copy this and paste the lines into your file, then change the IP addresses accordingly.)

```
remote 192.168.1.3 {
    exchange_mode main;
    lifetime time 24 hour;
    dpd_delay 10; //Set the dead peer detection timer
    proposal {
        encryption_algorithm 3des;
        hash_algorithm md5;
        authentication_method pre_shared_key;
        dh_group 2;
    }
}

sainfo address 191.168.1.2 any address 192.168.1.3 any {
    lifetime time 1 hour;
    encryption_algorithm 3des;
    authentication_algorithm hmac_md5;
    compression_algorithm deflate;
}

sainfo address 191.168.1.3 any address 192.168.1.2 any {
    lifetime time 1 hour;
    encryption_algorithm 3des;
    authentication_algorithm hmac_md5;
    compression_algorithm deflate;
}
```

Save your changes and quit the editor.

4. For the endpoint configuration defined in step 3 to take effect, the file `/etc/racoon/remote.conf` must be included in `/etc/racoon/racoon.conf`. Open `/etc/racoon/racoon.conf` for editing, and search for `include remote.conf`. If it is commented out (that is, if the line starts with a `#` sign), delete the `#`.
  5. While you have `/etc/racoon/racoon.conf` open, search the entire file for the string `aes` and remove it from any line that it appears in. Don't delete any entire line; just delete `aes` from it.
- Save any changes you made, then quit the editor.

6. You have now completed configuration activities for IPSec support on the proxy. To activate the services with the new configuration, start racoon by entering:

```
/etc/init.d/racoon start
```

7. You may now log off of the proxy server.

## How to test your IPSec implementation

To test the configuration you just completed:

1. With racoon running on both systems, log onto the iServer.
2. Ping the interface on the proxy, by entering:

```
ping -I interface 192.168.1.2
```

In the command, interface is the *name* of the local interface onto which you configured the IP address (192.168.1.3), for example, eth2. You can obtain this name from the `ifconfig` command.

3. If the IPSec is configured correctly, you will receive replies to your ping requests.
4. To verify that the pings and replies are traveling the secure IPSec channel, capture them with `ethereal` (or `tethereal`). The presence of ESP headers proves that the frames have traveled over the secure channel.

# **Part 2:**

## **VoIP Services**

## MSX Realms

*Realm-based routing*, or *RBR*, is supported by the MSX. This feature makes possible seamless VoIP telephony, transcending the usual concepts of *private* and *public* networks.

In IP networking, it is possible to segment an IP network so as to create IP address ranges (network addresses) that are private (that is, not directly addressable from outside their IP gateway). In addition, some IP address ranges are designated by Internet Assigned Numbers Authority (IANA) as “non-routable,” thereby allowing those network numbers to be used multiple times across enterprises and within an enterprise, without concern for duplication.<sup>1</sup>

But this scheme poses a challenge. A single enterprise can take advantage of this technique to set up two or more subnets with the same, or overlapping, non-routable network addresses. Now suppose they add some VoIP endpoints (H.323 terminals, or SIP user agents) on both of those segments, with the same IP address, or even a VoIP endpoint on one network with the same address as, say, a PC on another. How does a call from outside the enterprise, or for that matter even outside that network, get directed to the correct H.323 terminal or SIP user agent?

### **NexTone's Solution**

The MSX makes call routing through private network domains possible, through “realm-based routing” (RBR). The concept of realm-based routing is a transport-layer service that provides signaling and media services, including NAT for media, for endpoints either registered in, or statically defined as being in that realm. A realm appears as a unique signaling interface with an RSA (realm signaling address) and media routing interface with an RMA (realm media address) for reaching all these endpoints from outside the realm. No other address on the iServer (or in any other realm defined on the iServer) is directly exposed to these endpoints.

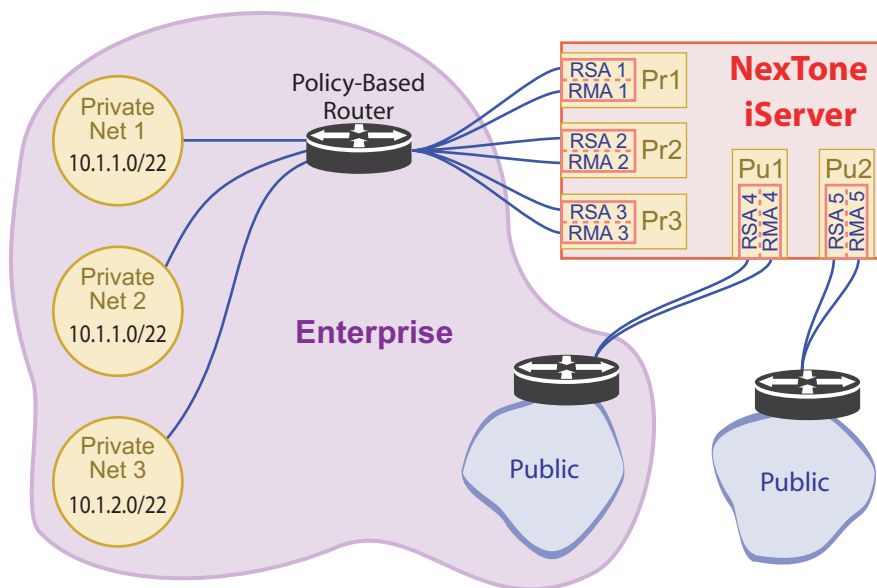
---

1. Addresses beginning with certain IANA-designated numbers, such as 10 or 192.168 are set aside by the IANA for this purpose, but any network, regardless of subnet size or address, can be made private by setting up its gateway routing appropriately.

## Realm-based Voice over IP

To facilitate routing across all networks, whether entirely public or entirely private, or a mixture of public and private subnets, MSX includes the concept of a *realm*. Figure 18 illustrates one example of a VoIP deployment using realms.

Figure 18. Realm Example



In Figure 18, the enterprise has three private networks, all of which have the same 22-bit CIDR prefix, and a public network. Key to realm-based services is the IP routing technique, supported by some routers, known as *policy-based routing* (PBR). PBR techniques enable a router to differentiate between packets to/from network endpoints based not only on the IP address of the destination device, but also on additional characteristics<sup>2</sup>.

The policy-based router is able to keep traffic to/from each device on all these networks separate, even if they have the same complete IP address, because the IP address and realm identification taken *together* constitute an endpoint's complete logical address. Under RBR, each private network is addressable by the MSX, and becomes a separate MSX *realm*.

2. In the case of MSX RBR, it is the packet's source IP address that is considered, in both signaling and media routing.

Note that each of the realms has either one or two corresponding physical interfaces on the session controller: an **RSA** (*realm signaling address*) interface for signaling, and an optional **RMA** (*realm media address*) interface for media routing. Signaling and media physical interfaces are kept separate to ensure that performance is not compromised. Each realm must have its own public or private logical IP address, unique on the MSX for its RSA. It may have additional logical addresses for its RMAs, and each RMA should be on its own physical interface. There is no advantage to assigning multiple logical RMA IP addresses for one realm, to a single physical interface.

Be aware that the physical interfaces are needed only for purposes of plumbing the logical addresses for the RSAs and RMAs. Physical IP addresses should never be used as realm signaling or media addresses, because systems using redundancy use must use logical IP addresses to pass realm services to the backup machine in event of an active processor failure.

*Note: The private networks shown on the MSX in Figure 18 are logical; all the RSAs and RMAs may reside either on the same two physical interfaces (one for RSA and one for RMA), or across multiple physical interfaces to ensure adequate call-carrying capacity.*

Multiple realms may share a common signaling *physical* interface on the MSX, but each realm will still have its own unique *logical* RSA. These separate logical addresses are the means by which the policy router, and thereby the MSX, distinguishes between the realms.

### **Endpoint registration flow**

In order for incoming calls to be received by an endpoint in a private realm, the endpoint passes its telephone number to the MSX during registration. Then the MSX saves this along with its logical address (RSA [IP] + NAT) information, which it passes to the router for that realm, to use when placing an incoming call. The router then uses the address information to forward that call data to the correct endpoint. Figure 19 shows the relationship between the endpoint and the MSX, when traffic originates at the endpoint. The registration process follows the same overall flow as an outgoing call, except that the end result is a registered endpoint, rather than the setup of an outgoing call.

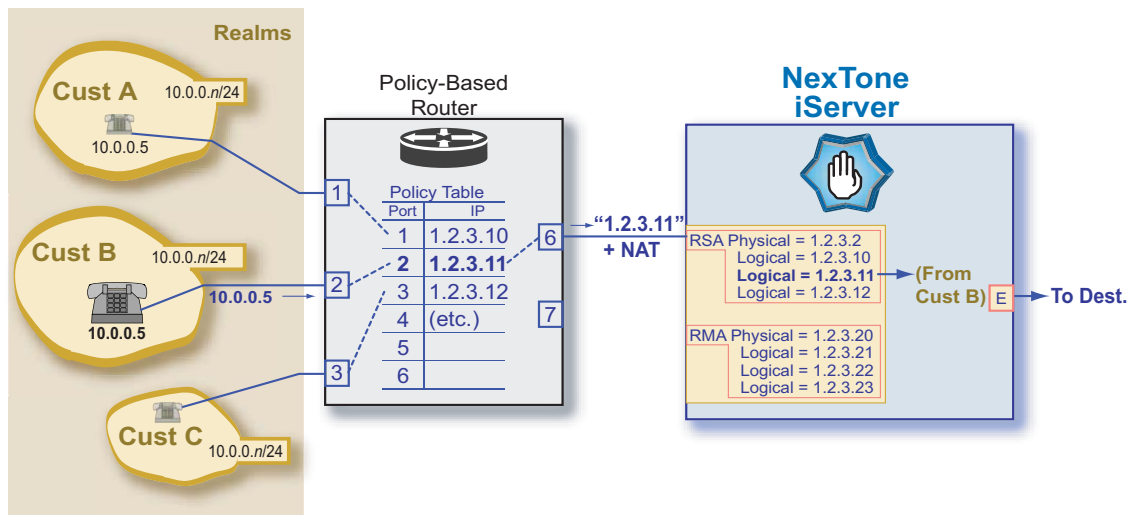
### **Outgoing call RBR flow**

Figure 19 gives an example of RBR outgoing call signaling flow. Here, a call originates from within a private network, and is sent to the destination



endpoint. The recipient can be either in the public domain or on another private network realm controlled by the MSX.

**Figure 19. Outgoing Call Signaling Example**



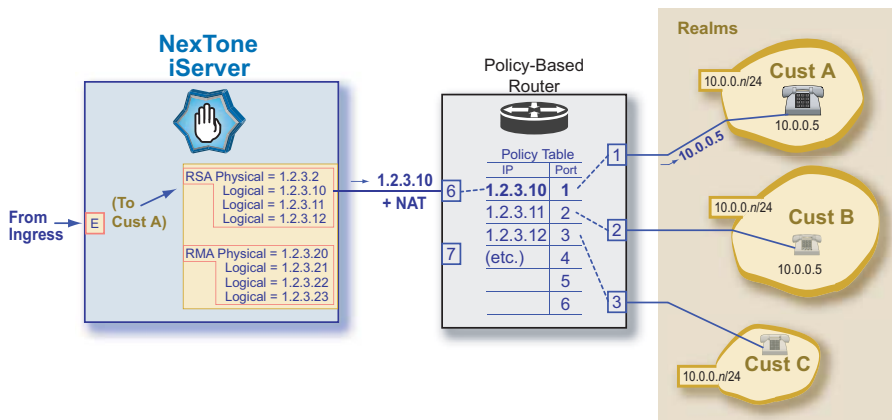
In the above example, two different customers (Cust A, and Cust B) each have a VoIP endpoint with the same 32-bit IP address (10.0.0.5), on a separate private LAN (10.n.n.n). The call proceeds as follows:

1. A user in **Customer B's** realm goes off hook from an IP phone at private address 10.0.0.5, and dials a number.
2. His LAN forwards the signaling packets to the policy-based router, which matches the port the call setup came in on (**port 2**) with the destination IP address (1.2.3.11, Customer B's RSA) in its policy-route table.
3. The router then forwards the request message packets to the MSX's logical RSA of 1.2.3.11.
4. The MSX, upon receiving the call setup request at logical IP 1.2.3.11, sets up the egress endpoint with NAT information telling it where to send the returning signaling traffic from that gateway.
5. Once the call is set up, media traffic flows in a similar fashion, through the RMA for that realm, until one or the other party terminates the call.

### Incoming call RBR flow

Figure 20 illustrates the flow of signaling in an incoming realm-routed call. In this example, a call originates from outside of the iServer's control.

**Figure 20. Incoming Call Signaling Example**



In the above example, a call from outside the MSX's control enters the network, destined for a telephone in Customer A's private realm. The call proceeds as follows:

1. A user outside the MSX's jurisdiction places a call to an endpoint in a realm within the MSX's jurisdiction.
2. The call arrives at the MSX from a registered gateway. The MSX matches the destination number with the RSA for the realm from which the endpoint registered, and forwards the call to the appropriate policy-based router, along with the NAT information it received for that endpoint when the endpoint registered.
3. The router then forwards the request to the endpoint possessing the telephone number dialed by the call originator.
4. The remainder of messages required to set up the call follow the same process, until the call is set up.
5. Once the call is underway, media traffic flows in a similar fashion, through the RMA for that realm, until one or the other party terminates the call.

### VLAN support

The MSX's support of 802.1q VLANs is presented in the chapter, "VLAN Support", on page 452.

## RBR deployment requirements

To deploy realm-based routing, you must have:

- An MSX of either version 2.1 or of 3.1 or later
- A feature license for FCE
- One or more gateway routers that support policy-based routing to the realms, if the realms have overlapping network IP address spaces. Otherwise, a gateway router that doesn't support PBR is sufficient.
- An MSX platform with at least one pair of physical interfaces, one for RSAs, and one for RMAs, in addition to any other interfaces required, such as for administrative traffic, or for the direct connection between primary and backup machine control interfaces on systems running redundancy.

## MSX RBR elements

NexTone's approach to routing across domains (public and private) uses two new classes of configuration element: *Realms* and *Pools*. Realms are described in *Realm-based Voice over IP* on page 175; Pools are described in *Setting up media devices and media routing pools* on page 297. When implementing an MSX, *first* create your pools, *then* your realms (since you can't complete the realm configuration process without the pool(s) for that realm being already in existence).

### ***Realms and their Creation***

A realm is defined and controlled by the signaling and media attributes listed and described in Table 12 and Table 13, below.

Table 12. Signaling Realm Attributes

Attribute	Description
<i>Note: Attributes marked with an asterisk (*) are required.</i>	
*Realm ID (hidden)	<p>An automatically-generated integer, unique for each realm. Once created, it cannot be changed, nor viewed in RSM Console. This ID is internally stored by the iServer on endpoints to improve lookup performance.</p> <p><i>Note: Device-to-realm assignments are based on this internal ID, not the Realm Name (below). Deleting a realm, then re-creating it with the same name, will break all the connections between the realm and endpoints assigned to it, since the new realm, while having the same name, will have a different Realm ID.</i></p>
*Realm Name	This is the key administrative attribute assigned when the realm is created, and used for editing and/or deleting realm attributes thereafter. Three reserved names have a special meaning: <i>default</i> , <i>any</i> and <i>none</i> .
*Realm Signaling Address (RSA)	This is the logical interface provided by the iServer as the peer for signaling for all endpoints within the domain.
*Realm Signaling Address Mask	The IP subnet mask for the RSA.
*Status	<p>Checking this checkbox makes this realm fully operational by enabling signaling. A realm must be disabled in order to change its RSA or Pool ID. This parameter is set via the Modify Realm panel's Enable Signaling checkbox, or the <b>admin</b> option for the <b>cli realm edit</b> command.</p> <p><i>Note: Disabling a realm while the MSX is running switches off signaling for that realm. New calls cannot be set up, but in-progress calls will continue until terminated.</i></p>
Signaling Vnet	Signaling Vnet for the realm.

**Table 13. Realm Media Attributes**

Attribute	Description
<i>Note: These attributes are required for media-routing (DMR) realms only.</i>	
Media Pool ID	This is the pool ID used to open NAT translations for media routing calls in this realm.
Intra-Realm Media Routing (imr)	This parameter enables or disables media routing for calls between endpoints in this realm.
Inter-Realm Media Routing (emr)	This parameter enables or disables media routing for calls between this realm and other realms.
Address Type	Public or Private. Indicates the addresses in this realm are “public” or “private”; see “Public realms and private realms” below.

**Public realms and private realms**

In general, a realm does not need to have any knowledge of *public* or *private*, since by definition it acts as a peering end for all endpoints within it. However, this information is used as a default for endpoints inside the realm. For an endpoint whose realm is “any”, the default of the realm always overrides the endpoint’s specific media routing policy.

**Endpoint rules**

Each endpoint participating in RBR must be assigned either to a specific realm or to the *any*<sup>3</sup> realm.

- To provide mobility, the realm name *any* means that the endpoint can dynamically register from any realm, and the iServer can route calls to it. In the MSX database, a realm association is optional for an endpoint, but if the realm name is blank, the iServer will not route calls to it, and will not let the endpoint register. Choosing *any* allows the endpoint to be statically registered with the MSX, and to specify its realm when it contacts the MSX. In this case, the iServer will not change the endpoint’s realm to a different name, but internally keep track of the endpoint (through its realm id).

---

3. Note that the realm called “any” should not be used for gateways, which do not register, and therefore must be defined as being in a specific realm.

- Realm name *none* means that no realm has been assigned to the endpoint. Such an endpoint cannot register or participate in calls.
- The *default* realm is a special realm, which exists only logically on the iServer.

The MSX initially assigns all endpoints (either created fresh or as a result of a system upgrade from a release not supporting RBR) to this realm by default. This realm can be disabled in the *servercfg* table.

Note that the *default* realm:

- Does not have a name
- Has a realm ID of 0
- Has an RSA of 0.0.0.0
- Is a *public* realm
- Has intra-realm media routing set to *don't care*.

### **SIP considerations**

The iServer does not need any global SIP domain configuration in releases supporting RBR (2.1, and 3.1 and subsequent). Each realm acts as an independent SIP domain and incoming calls from the realm must be addressed to the RSA.

### **Redundancy**

Redundancy is supported on MSXs running DMR. Physical IP addresses should never be used as realm signaling or media addresses, because systems using redundancy use must use logical IP addresses to pass realm services to the backup machine in event of an active processor failure.

### **The Add Realm dialog**

RSM Console provides a facility for creating realms and modifying their parameters. Figure 21 shows the Add Realm dialog. “Within realm media routing” is sometimes referred to as “internal media routing,” or *imr*; and

“between realms media routing” is sometimes referred to as “external media routing,” or *emr*.

**Figure 21. RSM Console’s Add Realm Dialog Box**

**Add Realm**

Partition: **admin**

☒ Enable Signaling

Realm Name:

SIP Authentication: **<none>**  
 inv  
 reg

CID Block:

CID Unblock:

☐ Signaling Only

**Signaling**

Realm Signaling Address:

Subnet Mask:

Vnet Name: **UNASSIGNED**

**Media**

Media Pool ID: **0**

Between Realms Media Routing: **Don't Care**

Within Realm Media Routing: **Don't Care**

**Mirror Proxy**

Registration ID: **1**

Port: **1**

☐ Public ☐ Private

**Add** **Cancel**

The fields for this panel are described in Table 14.

**Table 14. Modify Realm Panel Fields**

Field	Description
Partition	Select the partition to which the realm will belong.
Enable Signaling	Select this to enable new call set ups, or clear it to prevent new call set up. Clearing it blocks new call set-ups, but does not drop in-progress calls.
Realm Name	The name of the realm, alphanumeric, up to 31 characters.
Sip Authorization	The message types subject to authentication. See Table 57 on page 375 for possible values and their definitions.
CID Block <sup>a</sup> [SIP]	The prefix digit sequence a caller on this realm can send to prevent caller ID data from being sent for that call.
CID Unblock [SIP]	The digit sequence a caller on this realm can prepend to the dialed number, to send caller ID data for that call.
Signaling Only	Select this to disable media routing for the realm. The fields in the Media section are disabled when this checkbox is selected.
Realm Signaling Address	The RSA this realm uses.
Subnet Mask	The subnet mask for the RSA, in dotted-decimal format
Vnet Name	The name of the signaling Vnet to assign to the realm.
Media Pool Id <sup>b</sup>	The numeric identifier for the media pool this realm uses.
Within Realm Media Routing	The setting (Don't Care, Always On, Always Off, or On) to enable or disable calling from endpoints in the realm to other endpoints in the same realm. (See Table 15.)
Between Realms Media Routing	The setting (Don't Care, Always On, Always Off, or On) to enable or disable calling from endpoints in the realm to endpoints in other realms. (See Table 15.)
Registration ID [SIP]	Only for the MSX's "Mirror Proxy" realm, the regid of the far-end (real) proxy this realm will use.



Field	Description
Port [SIP]	The port number on the far-end proxy to which traffic originating in this realm will be sent.
Public / Private	These radio buttons designate this as either a public realm or a private realm.
Add button	Creates the new realm.
Cancel button	Closes the panel without applying changes.

- a. Both CID Block and Unblock are overrides to the default for the endpoint, set in RSM Console using Modify → Protocol → SIP Configure → Caller Id Block (check or uncheck).
- b. Media fields are only required for realms used for media routing.

#### **Internal and external media routing settings**

The settings shown in Figure 21 for **Within Realm Media Routing** (or *imr*, also known as “intra-realm” media routing) and **Between Realms Media Routing** (or *emr*, also known as “inter-realm” media routing) have the effects listed in Table 15.

Table 15. Realm Media Routing Settings

Orig Realm	Always on	On	Don't Care (xxx)	Always off
Dest Realm				
Always on	Media routed	Media routed	Media routed	Media routed
On	Media routed	Media routed	Media routed	Media <i>not</i> routed
Don't Care (xxx)	Media routed	Media routed	Media <i>not</i> routed	Media <i>not</i> routed
Always off	Media routed	Media <i>not</i> routed	Media <i>not</i> routed	Media <i>not</i> routed

You can think of the endpoint settings in the table in the following terms:

**Always on** – To communicate with me, you *must* route media.

**On** – I'd prefer to route media, and unless you specifically refuse (i.e., you're set to **Always off**), I will.

**Don't Care** – If you express a preference to route media (i.e., you're set to **Always on** or **On**), I will accommodate; otherwise, I won't route media.

**Always off** – I'd rather *not* route media, and unless you absolutely require it (i.e., you're set to **Always on**), I will not.

#### **Bulk assignment of endpoints to realms**

When a system is first set up, or upgraded to a release that supports realms from one that did not, every endpoint defined on that system must be assigned to a realm. By default, all existing endpoints are assigned to the default realm, but this is not where they need to be to process calls. In pre-RBR releases, every endpoint was defined as being either private or public, and only a single private network was supported by an MSX.

NexTone Support can provide a perl script, called `assignRealm.pl`, for assigning blocks of endpoints to a realm. In pre-RBR systems, endpoints may exist in either *public* or *private* address spaces. The syntax for the script depends on which kind of space you're assigning the endpoints from, as given in the following sections.

#### ***Using `assignRealm` on private endpoints***

To assign all *private* endpoints on a network at (for example) 192.168.180.0/24 to a realm named `Private-1`, enter the command:

```
assignRealm.pl -r Private-1 -i 192.168.180.0 -m 255.255.255.0
```

Where:

- `-r Private-1` names the realm to which endpoints are being assigned
- `-i 192.168.180.0` is the address of the endpoints' existing, private network
- `-m 255.255.255.0` is the network mask for the above network

After this command is run, all endpoints formerly defined as being “private” will be defined as being in the realm named `Private-1`. Of course, once the bulk assignment is completed, individual endpoints may be reassigned to any other defined realm name, using CLI or RSM Console.

#### *Using assignRealm on public endpoints*

To assign all *public* end points to a realm named `Public-1` for example, enter the command:

```
assignRealm.pl -r Public
```

Where:

- `-r Public-1` names the realm to which endpoints are being assigned

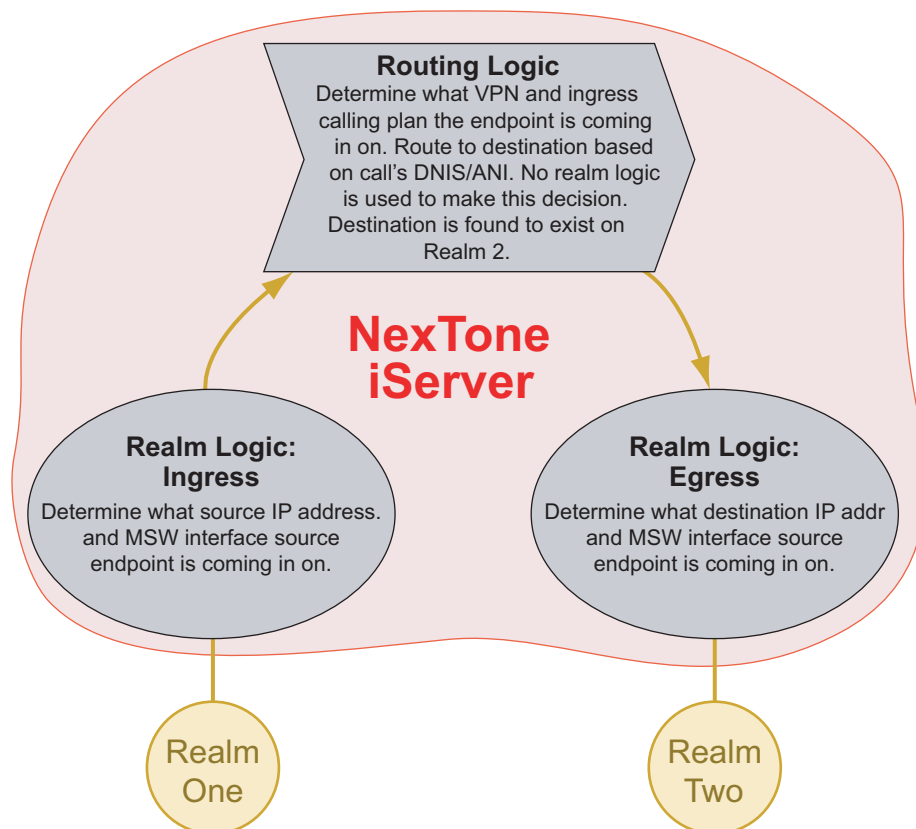
After this command is run, all endpoints formerly defined as being “public” will be defined as being in the realm named `Public-1`. Of course, once the bulk assignment is completed, individual endpoints may be reassigned to any other defined realm name, using CLI or RSM Console.

## Closed realms: E.164 VPN support

A realm is essentially an endpoint-access control mechanism. This does not mean that a realm provides any private E.164 space or any other private

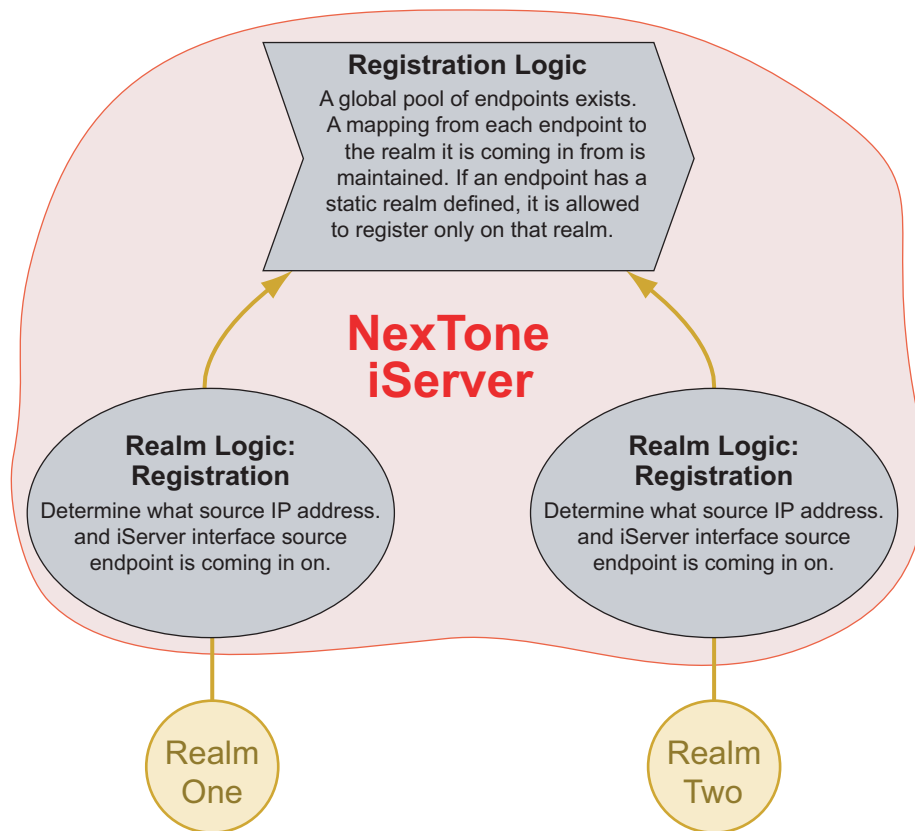
naming mechanisms. Figure 22 illustrates the MSX's call routing logic for realms.

**Figure 22. Realm Call Routing Logic**



E.164 aliases, trunk group ids, H.323 ids of endpoints are thus independent of which realm they are registering through or defined statically on. For example, 555 can only refer to one endpoint on the iServer, no matter what realm it exists on. Figure 23 illustrates this.

**Figure 23. Realm Registration Logic**



A realm can essentially be *closed*, by defining VPNs on endpoints which are in the realm. For example, an endpoint registering with an iServer, with E.164 alias 1001 in VPN 555, is interpreted by the system as 5551001. This allows multiple endpoints to register with the same E.164 alias. Calling between two endpoints in different VPNs is accomplished by dialling the VPN id 555 followed by 1001. A *VPN Group* is a group of VPNs that are allowed to call each other through this short-hand notation. Endpoints outside the VPN group must call the endpoints with an even longer E.164 number, for example 13015551001. In this case 1301 is referred to as the *VPN prefix*.

This VPN functionality is available only as with E.164 prefixes. H.323 IDs, trunk groups, etc., must be unique to the iServer across all realms.

## CLI commands for realm-based routing

Table 16 below lists the command-line interface (CLI) commands that may be used in configuring realm routing. General information on using the command line interface is given in the chapter, *The Command Line Interface* on page B-1.

Realms may be set up and maintained using RSM Console's **Realms** utility. See RSM Console's online help for more information.

Please note that there are some precautions to observe, with respect to realms, when performing command-line operations:

- These commands should be executed only while the MSX *is running*.
- If an MSX database containing new realms is put in service by executing `cli db create` followed by `cli db switch`, the realms will not be “plumbed” until the MSX is stopped and re-started.
- If an existing realm is deleted using `cli realm delete`, and the realm then is added back, each endpoint previously associated with it must be re-assigned to the realm. To do this, assign the endpoint to realm *none*, with:

```
cli iedge edit regid port realm none,
```

...and then with the re-added realm with:

```
cli iedge edit regid port realm realm name
```

**Table 16. Commands Related to Realms**

Task	Command	Options
List existing realms	cli realm list	
Add new realm	cli realm add <u>new realm name</u>	
Change existing realm	cli realm edit <u>realm name</u> [options] <sup>a</sup>	addr [public   private] admin [enable   disable] default [enable   disable] emr [ on   alwayson   alwaysoff   xxx <sup>b</sup> ] imr [ on   alwayson   alwaysoff   xxx] ifname <u>interface name</u> mask <u>rsa subnet mask</u> medpool <u>media pool ID</u> rsa <u>realm signaling address</u> sigpool <u>signaling pool ID</u> <sup>c</sup> vnetname <u>vnetname</u>
Delete an existing realm	cli realm delete <u>realm name</u>	
Assign endpoint to realm	cli iedge edit <u>regid port</u> realm [ <u>realm name</u>   none]	none dissociates the endpoint from all realms.

a. Options are entered as [name value] pairs, without intervening characters.

For example: cli realm edit realm name addr public to set the address type to "public."

b. xxx is "don't care" (see *Internal and external media routing settings* on page 185 for an explanation of these options).

c. The signaling or media pool ID is the integer value associated with each pool; it's the number following the word "Pool" on the FCE tab of the MSX Configuration window., or the *Pool Id* = "1" entry in *pools.xml*.

### **Generalized realm set-up procedure**

The basic procedure for CLI to set up realms is given below. Note that pools cannot be set up via CLI, so pools to be used must exist before beginning this procedure.

1. Log onto the iServer and change to the `/usr/local/nextone/bin` directory.

2. List existing realms using the `cli realm list` command, if desired.
3. Add, change or delete realms, as desired using the `cli realm add`, `edit` and `delete` commands.
4. Assign the realms to a pool (and set any other desired options) using the `cli realm edit` command.

### ***Output of realm list***

The `cli realm list` command produces a list of parameters for each realm defined on the MSX. A sample of the output is:

```

Realm zippo/1150270485
  Rsa          5.4.3.2
  Mask         255.255.255.0
  Adm Status   DISABLED
  Oper Status  UP
  Sig Poolid   1
  Media Poolid 2
  Addressing Type Private
  Inter-Realm MR Don't Care
  Intra-Realm MR Don't Care
  Main Interface znb0
  Virt Interface znb0:2

```

### **Output interpretation notes**

Most of the fields are self-evident, but please note the following:

- `zippo` in the sample above is the realm's textual name. The long integer following it is a system-generated internal realm ID to ensure the realm has a unique database key.
- `Adm Status` is the realm's administrative status, set using the `admin` parameter (or the **Enable Signaling** checkbox in RSM Console's **Modify Realm** panel). `DISABLED` means the realm is in the administrative state (i.e., it *is not* setting up new calls).
- `Oper Status` tells whether the entire realm is operational or disabled.
- `Inter-Realm MR` gives the `emr` parameter's setting for cross-realm media routing.
- `Intra-Realm MR` gives the `imr` parameter's setting for in-realm media routing.
- `Main Interface` names the Unix-level physical interface this realm's RSA resides on.



- `Virt Interface` tells which physical connection the network connection is plugged into. If there is more than one connector on the network interface card, the number following the colon (:) character indicates which connector services this realm's RSA.

## SIP Services

*Note: Running SIP services on MSX requires a feature license. See Chapter 5 for details on MSX licenses*

### SIP terminology

According to the standard, SIP *intermediate* systems are known as Proxy servers. They can be of the following types:

**Stateless Proxy server:** This type provides call setup services and does not stay in the call signaling path during the call. CDRs cannot be generated when running in this mode.

**Stateful Proxy server:** This type stays in the path for all *signaling* messages, and could be either *call* stateful or *call* stateless.

**Back-to-Back User Agent:** The back-to-back user agent (also called a “B2BUA”) is a call-stateful server that terminates and generates SIP call signaling.

**Registrar:** All registration messages from endpoints go to registrars, which keep track of the availability of the endpoints. In OBP and mirror proxy modes, the registrar is a server separate from the MSX.

### MSX as a SIP server

The MSX has the following SIP elements built into it:

**Back to Back User Agent:** In this mode, the MSX acts as a back-to-back user agent (B2BUA) in a SIP network for calls. All SIP calls must be destined to the MSX for the MSX to behave as B2BUA.

**Registrar:** The MSX processes all SIP REGISTER messages destined to it, and based on configured authentication, accepts the endpoint that sent the REGISTER into the database.

**Outbound Proxy (OBP) server:** If the MSX receives a SIP REGISTER or SIP INVITE not destined to it, then the MSX automatically behaves as an Outbound Proxy (if so configured). In this mode, the MSX relays the

REGISTER or INVITE message to the appropriate upstream server or endpoint specified in the SIP URI, but changes the “Contact:” header to the MSX address. For detailed information on OBP functionality, see *SIP outbound proxy and mirror proxy* on page 227.

**Mirror Proxy server:** So-called “mirror proxy” functionality applies when the MSX receives a call setup *from* a proxy, that is destined for yet another server, not for the MSX itself. For detailed information on mirror proxy functionality, see *SIP outbound proxy and mirror proxy* on page 227.

### **Registration support**

The MSX supports endpoint registration services in accordance with RFC 3261. The MSX supports both clear text and digest-based authentication for incoming REGISTER and INVITE messages. Once an endpoint has registered, it normally sends keep-alive REGISTERs to its registrar. Such keep-alive messages may be throttled, based on igmp or system timeout settings. These and other registration specifics are described in *Endpoint registration* on page 224.

When operating in Outbound Proxy (OBP) mode, the MSX passes registrations on to a separate registrar server. See *SIP outbound proxy and mirror proxy* on page 227 for more information on OBP mode.

### **SIP over TCP support**

The MSX supports SIP over TCP as a signaling and call control option. This support overcomes message fragmentation introduced by UDP for SIP messages larger than the network MTU. The 3-way handshake necessary to establish TCP connections also reduces the risk of Denial of Service (DoS) vulnerabilities. The following details of SIP over TCP support summarize this capability:

- MSX simultaneously supports TCP and UDP transport protocols for SIP signal processing.
- MSX simultaneously supports TCP and UDP SIP endpoints with a SIP TCP-to-UDP bridging function.
- MSX allows endpoints to change transport during the course of a call and defines mechanisms to convey this change of transport. MSX performs this transport protocol switchover using the transport parameter in the contact header as appropriate.
- TCP transport support for SIP calls work with High Availability and Stateful Call Migration deployment modes.

- TCP SIP transport support works with the Interworking Function (IWF)

### ***SIP UPDATE support***

Support for RFC 3311 UPDATE that allows handling of bidirectional UPDATE messages and their corresponding final responses on a per leg basis.

Tables 17 through 20 list the extent of SIP support on the MSX.

**Table 17. Supported SIP Methods**

Method	Extent of Support	Notes
ACK	Full, receive and transmit	The MSX receives and transmits ACKs.
BYE	Full, receive and transmit	The MSX receives and transmits BYEs,
CANCEL	Full, receive and transmit	The MSX receives and transmits CANCELS.
INFO	Full, receive and transmit	A UA uses INFO to send signaling to an endpoint with which is has an established media session.
INVITE	Full, receive only	The MSX receives and transmits RE-INVITES.
NOTIFY	Full, receive and transmit	Supported only when iServMSXer is running in OBP mode.
OPTIONS	Minimal, receive only	If the MSX is operating in OBP mode, it forwards a received OPTIONS; otherwise, it forwards a 404.
PRACK	Full, receive and transmit	Used to acknowledge receipt of reliably transported provisional responses (1xx).
REFER	Full, receive and transmit	A UA uses REFER to request that <i>another</i> UA contact a URI or URL appearing in the <code>Refer-To</code> field.
REGISTER	Full, receive and transmit	The MSX can proxy REGISTER Requests
SUBSCRIBE	Full, receive and transmit	Used by a UA to establish a subscription for receiving notifications about an event. Supported only when MSX is running in OBP mode.
UPDATE	Full, receive and transmit	Used to modify the state of a session without changing the state of the dialog.

**Table 18. Supported SIP Responses**

Response	Extent of Support	Notes
100 TRYING	Full, receive and transmit	The MSX has received a request and is processing it.
180 RINGING	Full, transmit, transmit	The dialed phone is ringing and the local end is providing the ringing tone.
181 Forwarded	Minimal, receive only	
183 Session Progress	Full, receive and transmit	The dialed phone is ringing and the remote end is providing the ringing tone.
200 OK	Full, receive and transmit	The MSX can process 200 OK requests. The information it returns with the response depends on the method used in the request.
300 Multiple Choices	Receive only	The MSX can internally handle a 300 Multiple Choices, and make a new call to the first contact in the response, if there is one.
301 Moved Permanently	Receive only	The MSX can internally handle a 301 Moved Permanently, and make a new call to the first contact in the response, if there is one.
302 Moved Temporarily	Receives all, but transmits only when the MSX is provisioned as "redirect"	The MSX can internally handle a 302 Moved Temporarily, and make a new call.
305 Use Proxy	Full support	The MSX can internally handle a 305 Use Proxy, and make a new call.
380 Alternative Service	None	
400 Bad Request	Full, receive and transmit	The MSX returns this response when it could not process the request due to malformed syntax.

Response	Extent of Support	Notes
401 Unauthorized	Minimal, receive and transmit	The MSX returns this response when it requires the user to authenticate a 401 unauthorized request.
402 Payment Required	Minimal, receive only	When the MSX receives this response from the callee, it forwards it to the caller and terminates the call.
403 Forbidden	Full, receive and transmit	The MSX will not process 403 Forbidden requests, although it understands them. These requests should not be repeated.
404 Not found	Full, receive and transmit	The MSX returning this status indicates one of the following: <ul style="list-style-type: none"> <li>• It has definitive information that the user does not exist at the domain specified in the Request-URI</li> <li>• The domain in the Request-URI does not match any of the domains handled by the recipient of the request</li> <li>• The MSX has received an ARJ or LRJ from an upstream gatekeeper.</li> </ul>
405 Method Not Allowed	Receive only	The MSX returns this response when the method specified in the Request-Line is not allowed for the address identified by the Request-URI.

Response	Extent of Support	Notes
406 Not Acceptable	Minimal, receive only	The callee sends this response when the resource identified by the request is only capable of generating response entities which have content characteristics not acceptable according to the accept header sent in the request. When the MSX receives this response from the callee, it forwards it to the caller and terminates the call.
407 Proxy Authentication Required	Minimal, receive and transmit	The MSX returns this response when it requires the client to first authenticate itself with the proxy. The proxy must return a Proxy-Authenticate header field containing a challenge applicable to the proxy for the request response.
408 Request Timeout	Minimal, receive only	The callee returns this status when its server is unable to produce a response within a suitable amount of time. When the MSX receives this status from the callee, it forwards it to the caller and terminates the call.
410 Gone	Full, receive and transmit	The MSX returns this response when the requested resource is no longer available at the MSX and no forwarding address is known.
413 Request Entity Too Large	Minimal, receive only	The server is refusing to process a request because the request entity-body is larger than the server is willing or able to process.

Response	Extent of Support	Notes
414 Request URI Too Long	Minimal, receive only	The callee sends this response when its server refuses to service the request since the Request-URI is longer than the server is willing to interpret. When the MSX receives this response from the callee, it forwards it to the caller and terminates the call.
415 Unsupported Media Type	Minimal, receive only	The callee generates this response when its server refuses to service the request since the message body of the request is in a format not supported by the server for the requested method. When the MSX receives this response from the callee, it forwards it to the caller and terminates the call.
416 Unsupported URI Scheme	Minimal, receive only	The server cannot process the request because the scheme of the URI in the Request-URI is unknown to it.
420 Bad Extension	Minimal, receive only	The callee sends this response when its server does not understand the protocol extension specified in a Proxy-Require or Require header field. When the MSX receives this response from the callee, it forwards it to the caller and terminates the call.
421 Extension Required	Minimal, receive only	The UAS needs a particular extension to process the request, but this extension is not listed in a Supported header field in the request.
422 Session Timer Interval Too Small	Full	The MSX will reject any request having a Session-Expires field with an interval shorter than the minimum set in the Min-SE field.



Response	Extent of Support	Notes
423 Interval Too Brief	Minimal, receive only	The server is rejecting the request because the expiration time of the resource refreshed by the request is too short.
480 Temporarily Unavailable	Minimal, receive only	The MSX generates this response when it contacted the callee's end system successfully, but the callee was unavailable.
481 Call Leg/Transaction Does Not Exist	Full, receive and transmit	The MSX returns this status under three conditions: if it receives a BYE request that does not match any existing call leg, if it receives a CANCEL request that does not match any existing transaction, or if it receives an INVITE with a To tag that does not match the local tag value.
482 Loop Detected	Receive only	The MSX returns this status when it receives a request with a Via path containing itself.
483 Too Many Hops	Full, receive and transmit	The MSX returns this response when it receives a request that contains a Max-Forwards header with the value zero.
484 Address Incomplete	Minimal, receive only	The callee generates this response when its server receives a request with a To address or Request-URI that was incomplete. When the MSX receives this response from the callee, it forwards it to the caller and terminates the call.
485 Ambiguous	Minimal, receive only	The MSX generates this response when the callee address provided in the request is ambiguous.

Response	Extent of Support	Notes
486 Busy Here	Full, receive and transmit	The MSX returns this response when it contacted the callee's end system successfully, but the callee was not willing or able to take additional calls.
487 Request Terminated	Full, receive and transmit	The MSX returns this response when the request was terminated by a CANCEL (or a BYE) request.
488 Not Acceptable Here	Minimal, receive only	The MSX generates this response when it is able to contact the user's agent successfully, but it could not process the request to the specific entity addressed by the Request-URI. This occurs when some aspects of the session description such as the requested bandwidth, or addressing style are not acceptable.
491 Request Pending	Minimal, receive only	The request was received by a UAS that had a pending request within the same dialog.
493 Undecipherable	Minimal, receive only	The request was received by a UAS that contained an encrypted MIME body for which the recipient does not possess or will not provide an appropriate decryption key.
500 Internal Error	Full, receive and transmit.	The callee sends this response when its server encounters an unexpected condition that prevents it from fulfilling the request. When the MSX receives this response from the callee, it forwards it to the caller and terminates the call.

Response	Extent of Support	Notes
501 Not Implemented	Full, receive and transmit	The MSX returns this response when it does not support the functionality required to fulfill the request.
502 Bad Gateway	Minimal, receive only	The callee sends this response when its server, while acting as a gateway or proxy, receives an invalid response from the downstream server it accessed in attempting to fulfill the request. When the MSX receives this response from the callee, it forwards it to the caller and terminates the call.
503 Service Unavailable	Minimal, receive only	The callee generates this response when its server is currently unable to handle the request due to a temporary overloading or maintenance of the server. When the MSX receives this response from the callee, it forwards it to the caller and terminates the call.
504 Server Time-out	Minimal, receive only	The callee sends this response when its server does not receive a timely response from the server it accessed in attempting to process the request.
505 Version Not Supported	Minimal, receive only	The callee sends this response when its server does not support, or refuses to support the SIP protocol version that was used in the request message. When the MSX receives this response from the callee, it forwards it to the caller and terminates the call.
513 Message Too Large	Minimal, receive only	The server was unable to process the request since the message length exceeded its capabilities.

Response	Extent of Support	Notes
600 Busy Everywhere	Minimal, receive only	The MSX returns this response when it successfully contacted the callee's end system but the callee was busy and did not wish to take the call at that time.
603 Decline	Full, receive and transmit	The MSX returns this response when it successfully contacted the callee's machine but the user explicitly did not wish to or could not participate.
604 Does Not Exist	Minimal, receive only	The callee sends this response when its server has authoritative information that the user indicated in the To request field does not exist anywhere. When the MSX receives this response from the callee, it forwards it to the caller and terminates the call.
606 Not Acceptable	Minimal, receive only	The MSX returns this response when it successfully contacted the user's agent, but some aspects of the session description such as the requested media, bandwidth, or addressing style were not acceptable.

**Table 19. Supported Headers that MSX can Receive or Transmit**

Header	Receive	Transmit
Accept	No	No
Accept-Encoding	No	No
Accept-Language	No	No
Allow	No	No
Authorization	No	Yes
Call-ID	Yes	Yes
Contact	Yes	Yes
Content Encoding	No	No
Content Length	Yes	Yes
Content Type	Yes	Yes
CSeq	Yes	Yes
Date	Yes	Yes
Encryption	No	No
Expires	Yes	Yes
From	Yes	Yes
Hide	No	No
Max-Forwards	Yes	Yes
Min-Expires	Yes	Yes
Min-SE	Yes	Yes
Organization	No	No
Priority	No	No
P-Asserted-Identity	Yes	Yes
P-Preferred-Identity	Yes	Yes
Proxy-Authenticate	Yes	Yes

Header	Receive	Transmit
Proxy-Authorization	Yes	Yes
Proxy-Require	No	No
Record-Route	Yes	Yes
Refer-To	Yes	Yes
Referred-By	Yes	Yes
Replaces	Yes	Yes
Require	No	No
Response-Key	No	No
Retry-After	Yes	Yes
Route	Yes	Yes
Server	No	No
Session-Expires	Yes	Yes
Subject	No	No
Timestamp	Yes	Yes
To	Yes	Yes
Unsupported	No	No
User-Agent	Yes	Yes
Via	Yes	Yes
Warning	No	No
WWW-Authenticate	Yes	Yes

**Table 20. Compatibility with RFC2327 (Media Capabilities in SDP)**

Field	Supported	Notes
a	Yes	Session attribute line
b	Yes	Bandwidth
c	Yes	Connection information

Field	Supported	Notes
k	Yes	Encryption
m	Yes	Media name and transport address
o	Yes	Owner/creator and session identifier
s	Yes	Session name
t	Yes	Time the session is active
v	Yes	Protocol version

## Configuring the MSX for SIP

For the MSX to route SIP calls in the network, you must:

- Configure endpoints for SIP
- Configure MSX-level SIP parameters

### Configuring an endpoint for SIP

You can configure an endpoint for SIP using the RSM Console. Refer to the RSM Console online help file for a detailed procedure describing SIP endpoint configuration.

The MSX identifies an endpoint with two SIP-specific elements: the SIP URI and the SIP Contact. SIP URI identifies the endpoint to the outside world, and applies only to endpoints that originate or terminate calls (IP phones, fax machines, etc.). The SIP Contact is the “internal” location of a SIP device, and applies to all SIP endpoint types.

#### SIP URI

The URI (Uniform Resource Indicator) provides a globally-unique identity to call-originating or -terminating endpoints. The URI format resembles an email address: user@server/domain, or bob@xyzcorp.com for example.

A relationship exists between the identifier in this field, and the SIP server name configured in the MSX, such that if no domain is specified for a SIP URI endpoint, the call completes to name@SIP server name. That is, if the MSX configured SIP server name was xyzcorp.com, and the SIP URI configured for that SIP phone was bob, that telephone would be known implicitly as bob@xyzcorp.com.

A SIP URI has no real meaning for SIP proxies or gateways, but it is required for terminating endpoints.

If DNS is not configured at your site, you can use the MSX IP. For example, 3015551212@10.0.0.1.

**Note:** *The URI field is required on the MSX endpoint definition for the endpoint to register. If the URI field is not provided, a forbidden message appears, and your SIP UA must be configured to send the URI for registration, and not the E.164.*

The MSX supports two types of dynamic registration for SIP:

- Register by E.164. For example, 3015551212@msx.nextone.com
- Register by URI. For example, some-identifier@registry-server

The generic SIP URI format is

[username | telephone number] @ [msx domain | IP address]

### **SIP Contact**

This optional field is the internal address that MSX uses to set up calls. It is required for URI-based routing, since the realm is not publicly addressable. It identifies the endpoint within its network. The format is [user name | telephone number] @ [domain | IP address [:portnum] ]. If the domain is specified, a domain server supplies the IP address. portnum is optional.

The Contact field is either provided by the device during registration, or can be provisioned directly on the MSX. If provisioned for a gateway, the user portion of this field is irrelevant, and is omitted. If this field is specified as just an IP address (or domain) for a terminating endpoint, the MSX assumes the full Contact ID to be the user name from the SIP URI, at the IP address/domain given in this field. Therefore, simply specifying the IP address or domain is sufficient.

### **Dynamic registration support**

If DNS is not configured at your site, the IP of the MSX, for example, 3015551212@10.0.0.1, allows you to create dynamic registration support. The URI field must be filled on the endpoint definition on the MSX (using RSM Console Add → Endpoint → Generic IP Device → Protocol → SIP → URI) otherwise you won't be able to register—you'll get a forbidden message and your SIP UA must be configured to send the URI for registration, not the E.164.

### **Broadsoft redundancy support**

MSX transparently supports redundancy for Broadsoft SIP proxies. Provision one of the Broadsoft machines in the redundant set, and designate it as a



member of such a set. The relationship between the members of the cluster is automatically maintained by the Broadsoft clustering software.

#### ***RSM Console procedure***

1. When setting up the endpoint, select SIP Proxy from the **Provision an Endpoint** window's Device Type pull-down menu.
2. Fill in the remaining relevant information on that screen, and click on the Advanced tab.
3. From the Vendor pull-down list, select Broadsoft.
4. Click the Protocol tab.
5. In the URI (Sip/H323) field (highlighted in Figure 24), enter the fully-qualified domain name (FQDN) for the redundant set.

**Figure 24. Setting the SIP URI**

The screenshot shows the 'Provision an Endpoint' window with the 'Protocol' tab selected. The window is divided into three main sections: Gateway/Proxy, SIP/H323, and Trunk Group. The SIP/H323 section is currently active, showing checkboxes for 'SIP' (checked) and 'H.323' (unchecked). Below these is a text field labeled 'URI (Sip / H323)' which is highlighted with a grey oval. The Trunk Group section contains fields for 'Src. Trunk Group', 'Dest. Trunk Group', 'New Src. Ingress Trunk Group', and 'New Src. Egress Trunk Group', along with checkboxes for 'Send Dest. Trunk Group' and 'Remove Src. Trunk Group'. At the bottom are 'OK' and 'Cancel' buttons.

- Click the SIP Configure button. The **SIP Protocol Parameters** panel shown in Figure 25 appears.

**Figure 25. Enabling Broadsoft Redundancy**

- Select FQDN redundancy.
- When finished, click OK twice.
- Log onto the MSX as root, and activate the endpoint by entering:

```
cli iedge edit regid uport static 0.0.0.0
```

**Note:** *Setting the IP on a static endpoint to 0.0.0.0 forces a DNS lookup for that endpoint.*

### **CLI procedure**

If you prefer, you can issue CLI commands to set this feature as follows:

- If necessary, add the endpoint with:  

```
cli iedge add regid uport
```
- Declare the endpoint's type as SIP proxy:  

```
cli iedge edit regid uport type sipproxy
```
- Declare the endpoint's vendor as Broadsoft:  

```
cli iedge edit regid uport vendor broadsoft
```
- Assign the cluster's SIP URI to it:  

```
cli iedge edit regid uport uri sip uri
```

5. Enable or disable redundancy:

```
cli iedge edit regid uport redundancy [enable | disable]
```

6. Activate the endpoint (0.0.0.0 forces a DNS lookup):

```
cli iedge edit regid uport static 0.0.0.0
```

### **Setting signaling rate limits for non-INVITE messages for specific endpoints**

You can define a signaling rate limit that limits the number of non-INVITE request methods a SIP endpoint can receive. You can define such a limit by editing the endpoint configuration using the `cli iedge edit` command as follows:

```
cli iedge edit <regid> <uport> max_sip_noninv_dlgcount <value>
```

where:

<regid> and <uport> identify the endpoint you want to configure.

`max_sip_noninv_dlgcount <value>` specifies the maximum number of concurrent non-INVITE requests to allow at the endpoint.

For example, to define 100 as the maximum number of concurrent non-INVITE requests for an endpoint named “ep2,” use the following:

```
cli iedge edit ep2 max_sip_noninv_dlgcount 100
```

### **Setting incoming call forwarding transport protocol**

You can configure how incoming calls are transported while being forwarded using the following command:

```
cli iedge edit <regid> <uport> contact  
"<ip_addr>:<port>;transport=<tcp|udp>"
```

Setting the transport parameter to `transport=tcp`, forces the MSX transport layer to look for an existing TCP connection or to open a new TCP connection in the contact of the outgoing `<ip_addr>:<port>`. If no transport protocol is set, the default is UDP.

<regid> and <uport> identify the endpoint to configure

<ip\_addr>:<port> identify the contact IP address and port to be set to **tcp**

For example, to set the transport for an incoming call to tcp, you would create the configured command:

```
cli iedge edit ep1 contact "192.152.36.51:5060;transport=tcp"
```

**Note:** The contact value must be surrounded by double quotation marks (“ ”) as specified in the command example.

**Setting idle TCP connection timeouts for applicable gateways**

You can configure an TCP timeout that specifies when an idle connection should be closed. This configuration is applicable to gateway endpoints where connections must persist across transactions and dialogs. These TCP connections are removed from the MSX only when they time out. The default value for this configuration is 180 seconds, or 3 minutes.

```
cli iedge edit <regid> <uport> idle-tcp-connection-timeout <seconds>
```

For example, to set the timeout for an idle gateway TCP connection, you would create the configured command:

```
cli iedge edit epl idle-tcp-connection-timeout 60
```

**Configuring global SIP parameters**

You can configure SIP parameters on the MSX using either RSM Console or the sconfig utility (see *Global configuration using nxconfig.pl* on page 18). To configure SIP parameters using sconfig, follow the steps given below:

1. Log in to the MSX.
2. Enter:
 

```
nxconfig.pl -E
```
3. Repeatedly press <Enter> until reaching the following prompt:
 

```
SIP:
-----
sipminse [nnn]:
```

Set a value for the minimum session timer expiry value that the MSX will accept, in seconds, and press <Enter>. The default value is 600 seconds.
4. Next, set the SIP session timer expiry value that will be set by the MSX:
 

```
sipsess [nnnn]?
```

Set a value, in seconds, and press <Enter>. The default value is 3600 seconds.
5. The next prompt sets the maximum number of times a SIP request can be forwarded:
 

```
sipmaxforwards [nn]?
```

The default for this value in the shipped MSX is 70. Set a value and press <Enter>.
6. The next prompt enables or disables support for SIP authentication and, if enabled, sets the authentication method.

```
sipauth <local|radius|[none]>
```

**Note:** *Local authentication is not supported in this release.*

Enter radius to specify RADIUS authentication, or none to specify no authentication.

7. The next prompt enables/disables outbound proxy (OBP) mode. Note that the SIP Server Type (set in the next prompt) must be set to `proxystateful` for the MSX to operate as an outbound proxy.

```
obp <[0]|1>
```

Enter 0 to disable or 1 to enable OBP mode, then press <Enter>.

8. The next prompt sets the state in which the MSX operates when connecting a SIP call:

```
sipservertype[proxystateful]:
```

```
a) => proxystateful
```

```
b) => redirect
```

```
c) => proxy
```

```
q) => quit
```

Enter a letter for the server type or q to accept the current setting, then press <Enter>.

The options have the following meanings:

- 8.1 When configured in *redirect* mode (not currently supported), the MSX acts as a user agent server that generates 3xx responses to requests it receives, directing the client to contact an alternate set of URIs. When configured in this mode, the MSX does not generate Call Detail Records (CDRs) for the calls it connects.
- 8.2 When configured in *proxy* mode, the MSX acts as a logical entity that does not maintain client or server transaction states when processing a request. It forwards every request it receives downstream and every response it receives upstream. When configured in this mode, the MSX does not generate Call Detail Records (CDRs) for the calls it connects.
- 8.3 When configured for *proxystateful* mode, the MSX acts as a B2BUA entity that maintains the client and server transaction states when processing a request. When you configure the MSX in this mode, you can generate Call Detail Records (CDRs) for the calls that the MSX processes.

9. Press <Enter> repeatedly until the following prompt appears:

Do you want to commit the changes (y/n)? [y]

Press <Enter>.

10. At the following prompt:

Changes made require iServer restart (y/n) [y]:

Type *y* to restart the MSX processes. Note that in-process H.323 calls, all incomplete call setups, are dropped during a restart.

#### **Additional global SIP parameters defined in *servercfg***

Values assigned to the *servercfg* attributes listed in this section are edited using the *nxconfig.pl* utility. Be careful when editing these values. It is possible to enter values which, in a few cases, can produce undesired results in scripts and other programs that use these values. The following parameter descriptions are preceded by the *nxconfig.pl* command that sets the parameter.

To run these commands:

1. Log into the MSX.
2. Enter the following command:

***nxconfig.pl -e siptimer-T1 -v value***

The INVITE transaction consists of a three-way handshake. The client transaction sends an INVITE, the server transaction sends responses, and the client transaction sends an ACK. For unreliable transports (such as UDP), the client transaction retransmits requests at an interval that starts at T1 seconds and doubles after every retransmission. T1 is an estimate of the round-trip time (RTT), and it defaults to 500 milliseconds. The units used by *nxconfig.pl* are microseconds.

***nxconfig.pl -e siptimer-T2 -v value***

Non-INVITE transactions do not make use of ACK. They are simple request-response interactions. For unreliable transports, requests are retransmitted at an interval that starts at T1 and doubles until it hits T2. If a provisional response is received, retransmissions continue for unreliable transports, but at an interval of T2.

So, in essence, T2 is a limit on the interval-doubling effect upon T1, described above. The units used by *nxconfig.pl* are microseconds. The default is 4000000 (4 seconds).

**`nxconfig.pl -e siptimer-C -v value`**

The maximum time allowed for the MSX to receive a response numerically greater than 100 Trying, indicating that the call is proceeding. For a more-detailed description, see *SIP Timer C* on page 91.

The units used by `nxconfig.pl` are seconds. The default value is 180 (3 minutes).

**`nxconfig.pl -e siptimer-shorthunt -v value`**

This parameter specifies a time interval, in seconds, for the MSX acting as a B2BUA to wait before retrying an outgoing setup with a redundant endpoint. The units used by `nxconfig.pl` are seconds. The default waiting time for retry is 10 seconds.

**`nxconfig.pl -e siptimer-I value`**

This timer defines how long a server transaction can remain in the Confirmed state before it times out, and is terminated. In an MSX, it applies only when the MSX is running in OBP mode, when the proxy challenges INVITEs. An INVITE gets challenged by the iServer receiving a 401 or 407 from the remote proxy. Normally, the iServer cleans up the call upon receiving any final response (i.e., a SIP response  $\geq 200$ ) but not in the 401/407 case, where the iServer has to keep this call until the UA responds to the remote proxy's challenge.

If a UA for any reason doesn't respond to INVITE challenges (due to a malfunction, crash, etc.), calls on the iServer never get released, and continue to consume vports. *Siptimer I* protects against this case. After Timer I expires, the MSX cleans up hung calls.

The units used by `nxconfig.pl` are seconds. The default value is 5 seconds.

**`nxconfig.pl -e sipminse -v value`**

The minimum SIP session expiry (RFC 4028, Min-SE Header Field) timer, in seconds, that the iServer will accept from an endpoint's INVITE or UPDATE. Values smaller than this are rejected. The units used by `nxconfig.pl` are seconds. The default value for this parameter is 600 seconds.

**`nxconfig.pl -e sipsess -v value`**

The upper limit to the *session expiry* timer value (RFC 4028, Session-Expires header field) that the iServer will accept from an endpoint. Units are in seconds. The default value for this parameter is 3600 seconds.

`nxconfig.pl -e siptrans-invitec -v value`

This parameter controls the number of times an INVITE is retransmitted before the call setup attempt is abandoned. The default value for this parameter is 7. The MSX uses this count to simulate the Timer B defined in RFC3261.

`nxconfig.pl -e sipmaxforwards -v value`

The maximum number of forward hops a call may make during setup attempt before it is abandoned, as defined for the SIP Max-Forwards header. The default is 70. This value is included in the header, and gets decremented with each forward, until the call either completes, or this value reaches zero, at which time the setup is abandoned.

`nxconfig.pl -e sipqlen -v value`

Limits the number of pending SIP call setups. May be useful in combating denial-of-service (*DOS*), attacks. This value should not be altered except on direction of NexTone Support.

`nxconfig.pl -e sipdelay -v value`

Introduces a delay sometimes required with Polycom phones when using *shared call appearances* (during conferencing setup), while running in *OBP mirror proxy* mode. It delays INVITEs, and so should only be used when necessary, as directed by NexTone Support. The units used by `nxconfig.pl` are milliseconds. An initial value of 200 is recommended when tuning this parameter.

`nxconfig.pl -e use3261Branch -v value`

The default SIP “branch” parameter sent by the MSX lacks the preamble string specified by RFC 3261 (“z9hG4bK”). By enabling this feature, the MSX inserts the string, which some network elements require. Specify a value of 1 to enable or 0 to disable this feature.

`nxconfig.pl -e max-transport-mtu-size -v <bytes>`

The Network Maximum Transmission Unit (MTU) size command lets you establish an MTU for your network. RFC 3261 gives guidelines for MTU sizing stating that if a request is within 200 bytes of the path MTU, or if it is larger than 1300 bytes and the path MTU is unknown, the request must be sent using a congestion-controlled transport protocol like TCP. The default MTU value is 1300 for Ethernet networks. When a SIP packet is greater than **max-transport-mtu-size**, the MSX tries to send the packet over TCP transport. If there is a network failure, the packet is resent using UDP.

`nxconfig.pl -e disable-sip-over-udp -v <0/1>`

As required by RFC 3261, MSX supports a user agent that listens over both UDP and TCP protocols for incoming connections. MSX also allows you to



disable UDP on all realms, requiring all configured endpoints to connect to the MSX over TCP. When this configuration is enabled, all incoming and outgoing UDP traffic would be blocked on the MSX, preventing endpoints from connecting or receiving calls over UDP.

*usage:*

valid values: 1 (enable), 0 (disable)

default value: 0 indicating that MSX listens over both UDP and TCP

### **Other notes on SIP call processing**

#### **Max-forwards header support**

Support for the Max-forwards header is supported. The setting is global to the system, and can be configured in the `sipmaxforwards` attribute of `servercfg` using the `nxconfig.pl` utility. It can also be set in RSM Console in the Maximum Forwards field of the **iServer->Configuration->SIP** configuration page. When the system is installed, the default value is set to 70 in accordance with RFC 3261.

To set the `sipmaxforwards` value using `nxconfig.pl`, log into the iServer and enter:

```
nxconfig.pl -e sipmaxforwards -v value
```

#### **Diversion header support**

The SIP `Diversion` header should contain a diversion reason for diverted calls. This `reason` information, if present, is placed into CDR field 81. That field also provides the diverting device's realm name and SIP URI. See Table 15 on Page 396 for a description of CDR field 81.

Upon receiving a 3xx redirect SIP response, the MSX normally sends a new INVITE to the contact(s) specified in the 3xx redirect message. In the new INVITE, the MSX inserts a SIP Diversion header with the call diverter's information, when both of the following conditions are met:

- No Diversion header is present in the incoming SIP 3xx message
- The diverter is configured either as a Generic IP Phone or as a SIP Gateway.

If the MSX does insert a SIP Diversion header in the outgoing INVITE, then the call diverter's information is recorded in CDR field 81.

#### **3xx final response processing**

300 and 302 Final Response processing has been enabled on the MSX. No further configuration is required.

The MSX handles all 3xx responses by looking at the first contact field in the response and making a new call. Any other contact fields are ignored.

If the 3xx Final response contains a trunk group parameter (`cic=xxxx`), then the trunk group parameter is added to the outgoing SIP INVITE.

### **SIP access control from Via, Contact and From headers**

The IP addresses in the host portion of the URIs in the (topmost) `Via` header, `Contact` and `From` headers are now compared against provisioned entities in the MSX database, providing access control to requests/messages from intermediate entities such as Proxy servers and Back-to-Back User Agents (B2BUA).

### **Configuration for rfc2833**

There is now a global configuration option that allows for the negotiation of the rfc2833 codec capability. This is applicable only to IWF calls, and this parameter always conveys that the H.323 leg can support rfc2833. This forces rfc2833 codec on the SIP leg during terminal capability negotiation.

To enable this feature, log into the iServer and enter:

```
nxconfig.pl -e always2833 -v 1
```

The default value for this parameter is 0 (off).

### **Retry-after**

The SIP Retry-after header is supported.

### **Relaying of trunk group parameters**

If the incoming INVITE contains a trunk group parameter in the user part of the SIP URI, then this information is relayed on the egress SIP INVITE also, as the `dtg` parameter (destination trunk group), known internally to the MSX as `destinationCircuitID`. The `otg` (origin trunk group) parameter is found in the `Contact` field, known internally to the MSX as `sourceCircuitID`.

### **Support for an E.164 Leading Plus Sign**

The E.164 standard provides for prepending a plus sign, “+”, to a destination number string, to indicate that it is an E.164 standard number (i.e., international direct dialing is possible). MSX supports this feature for SIP calls (but not H.323 calls) as described here.

- The MSX supports a plus sign preceding the `user` portion of the `RequestURI`, `From` and `To` fields.
- The presence of the plus sign in the `RequestURI` and `To` fields is controlled on a per-endpoint basis. For each endpoint, an `Outgoing Prefix` may be defined (on RSM Console’s `Advanced` tab for that endpoint). When this outgoing prefix is defined as “+”, a “+” will be prepended to the `user` portion of the

RequestURI and To fields for all calls sent to that endpoint regardless of the presence or absence of the “+” in calls coming into the MSX. When the plus is *not* defined in the outgoing prefix, the RequestURI and To fields will *never* contain a leading plus, regardless of the value sent into the MSX.

- The user portion of the From is passed through unchanged. So if the plus sign is present in the From coming into the MSX, it will be present in the From sent from the MSX to the terminating endpoint.
- For this feature to function, the global parameter leading-plus-sign-in-uri must be enabled. You can do this using nxconfig.pl.

## SIP trunk group support

MSX supports trunk-based routing. Details on that capability are given in the *Calling Plans* chapter, under the heading, *iServer trunk group support* on page 503.

A SIP INVITE can include fields representing the source and destination trunk groups. How this data appears differs according to the exact implementation for a given network owner/operator. Table 21 shows two examples of how trunk group data transfer may be implemented by a network operator.

**Table 21. Example SIP INVITE Trunk Group Fields**

	Trunk Type	Field Name	Example
Example 1	source	otg	From: sip:12345@x.com; otg=TRUNKB
	destination	dtg	INVITE sip:12345@x.com; dtg=TRUNKA; user=phone SIP/2.0
Example 2	source	tgrp	Contact: sip:gateway1.someprovider.il.us; tgrp="local=1001BSTAOMA01MN"
	destination	tgrp	INVITE sip:+14085551212;tgrp=local=tg55/3@ someprovider.il.us

**Note:** When setting up trunking for SIP, the values entered in the *Modify (endpoint)* panel **must** include the field name and equals sign (=), as shown in the examples.

The source and destination fields listed in the table map to the MSX's Src. Trunk Group and Dest. Trunk Group fields, respectively, as shown on RSM Console's *Modify (endpoint type)* panel.

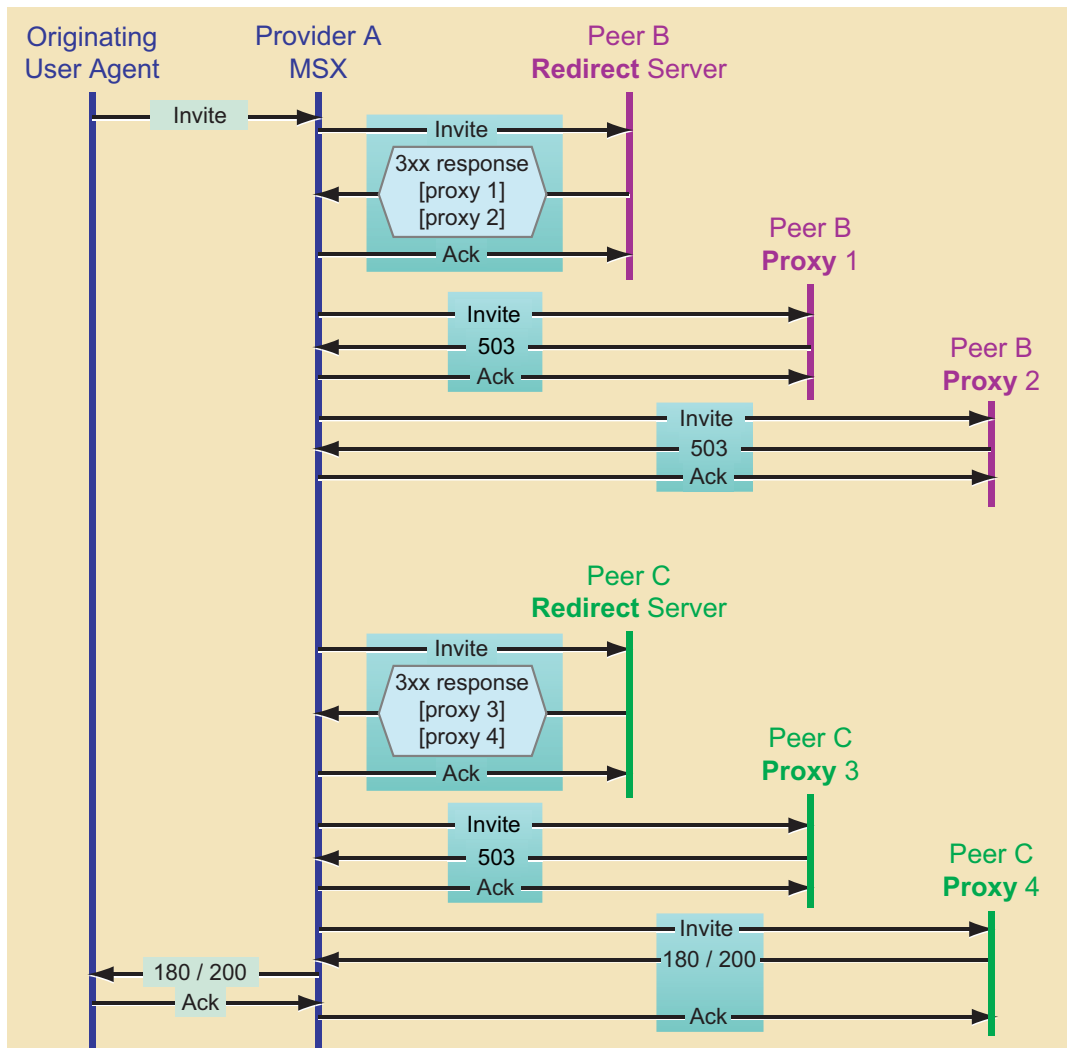
## SIP call flow

SIP call setup follows a sequence of steps prescribed by the SIP protocol definition. Good information on SIP and its components is available at <http://www.cs.columbia.edu/sip/overview.html>. That site also provides the complete protocol standard for those interested, at <http://www.faqs.org/rfcs/rfc3261.html>. The information in this section is not intended to constitute a complete discussion of SIP call flows, but to give a basis for help in understanding the rest of the chapter.

Note that SIP's allowable sequence of steps is less strictly defined than that for H.323. While each message (e.g., an INVITE or 503, etc.) must be acknowledged (note the *ACKs*), it is not always necessary for one step to complete before the next is begun. In this way, SIP is somewhat more free-form than H.323. The MSX supports this aspect of SIP.

Figure 26 depicts one simple SIP call setup scenario.

**Figure 26. SIP Call Setup with Multiple Addresses Returned**



In the scenario illustrated in this figure, three providers are involved, called Provider A, and Peers B and C. The sequence depicted is strictly followed, from top to bottom. The steps are:

1. Provider A's originating user agent sends a SIP INVITE to the MSX.
2. The MSX in turn sends an INVITE of its own to a SIP Redirect Server belonging to one of its peering partners, Peer B.

3. Peer B's redirect server sends back in its 3xx response one or more (in this example, two) addresses of SIP proxy servers, through which the call may be routed.
4. The MSX first tries to set up a call with Peer B's Proxy 1, and the setup does not complete (as shown by the 503 (service unavailable) message).
5. The MSX then tries to set up the call through Peer B's Proxy 2, but that attempt fails, too.
6. Now the MSX sends a new INVITE, this time to Peer C's redirect server, which returns the addresses of two Peer C's own proxy servers.
7. The MSX sends an INVITE to Peer C's Proxy 3, and again the call fails through this proxy, with the proxy returning a 503.
8. Finally, the MSX sends an INVITE to Peer C's Gateway 4, and the call gets connected, with a 180 (ring) and 200 (connect). With a final ACK from the originating UA, the call proceeds.

### ***SIP NAT traversal***

A special case for SIP call routing is SIP NAT Traversal. This section describes how the MSX handles this case.

#### **The problem**

SIP endpoints behind a NAT device (firewall/router/gateway, etc.) use private IP addresses in SIP signaling messages. SIP signaling messages cannot be sent to these devices from outside the NAT device unless port forwarding is set up on the NAT device.

#### **Our solution – overview**

##### ***Outbound calls***

The MSX detects that the SIP origination endpoint is behind a NAT device by detecting a mismatch between the IP address in the standard `Via` header and the source address in the INVITE. This release includes an option, called *automatic NAT detect*, that directs how the MSX handles this. It tells the MSX whether to use the address of the NAT device (`enable`) or use the existing procedure (`disable`) to communicate with an endpoint where the two addresses (`Via` and `INVITE`) don't match.

The NAT detect option is set using the `cli iedge edit` command as follows:

```
cli iedge edit regid uport natdetect [enable | disable]
```

If the NAT detect flag is enabled, MSX sends SIP signaling messages to the IP address and port from which it received the initial INVITE.

***Inbound calls***

Inbound calls require that the SIP endpoint be registered with the MSX, either dynamically or statically. This is described in *Endpoint registration* on page 224.

When the MSX has to send an INVITE to an endpoint, it uses the NAT IP and port from where endpoint registered. For dynamic or transient<sup>1</sup> endpoints, this information is obtained from the REGISTER messages.

**Operational summary**

This section details the net results of using various combinations of option settings and calling directions (inbound/outbound).

***Inbound calls***

Calls made to a user agent behind a NAT device will complete in either of the following two cases:

1. natdetect is set to enable, and the endpoint registers periodically with the MSX to keep the NAT binding alive, or
2. natip and natport are statically configured, and port forwarding is enabled on the router.

***Outbound calls***

Calls made from a user agent behind a NAT device will complete in either of the following two cases:

1. natdetect is set to enable, **or**
2. natip and natport are statically configured, and port forwarding is enabled on the router.

**Media routing**

Media routing will work for calls to/from a user agent behind a NAT device if:

- natdetect is set to enable

**Caveats**

Some points to note when implementing NAT traversal:

1. Multiple endpoints behind a single NAT device must be provisioned as multiple uports under the same regid.

---

1. A transient endpoint is one not in the database, which has policy applied to it based on the subnet from which it registered. See *Subnet CAC* on page 83 for more information.

2. The NAT device's NAT scheme must be symmetric for both signaling and media for our NAT traversal approach to work.
3. MSX does not do anything to keep the NAT binding alive, so the "keep alive" constraint mentioned under *Dynamic and transient registration* (below) must be observed.

### ***Endpoint registration***

To place or receive calls, endpoints must register with either the iServer (or an external registration server, through the iServer acting as its proxy). Registration can be *static* (permanently in the MSX's database, by its IP address), *dynamic* (in the MSX's database, but by regid, not IP address), or *transient* (only in the cache, with policy applied based on the subnet it's in, and not in the MSX's database).

#### **Static registration**

To effect static registration, configure a NAT IP and port for the endpoint on the MSX, and set up port forwarding to the endpoint on the NAT device.

The CLI commands to set up the NAT IP and port for the endpoint:

```
cli iedge edit regid uport natip IP address
cli iedge edit regid uport natport port
```

In RSM Console, these parameters are set on the **SIP Protocol Parameters** window. Be sure to disable Automatic NAT Detection there, to enable the NAT IP and NAT Port fields.

#### **Dynamic and transient registration**

Under dynamic and transient registrations, the endpoint registers periodically with the MSX (called *keep-alive*) to keep its registration active. See *Keep-alive behavior* on page 225. The MSX's default, global keep-alive interval is 900 seconds, but you can alter this global default value.

Two iServer parameters control time-out behavior:

- `age-timeout` attribute in `servercfg`, which defines the time interval for which a registered endpoint is considered active, and before which it must re-register.
- `cli igrp ... timeout`, which sets a time-out value at the iEdge group (igrp) level for igrp and subnet CAC control modes.

The protocol parameters by which endpoints communicate timeout values in SIP are `REGISTER ... expires`.

To set this timeout value, either:



- Edit the `age-timeout` value in `servercfg` by logging in to the iServer and typing:

```
nxconfig.pl -e age-timeout -v value
```

where value is the timeout period, in seconds;

- or -

- In RSM Console, set the global timeout to another value on the **iServerConfiguration** window, via its System —> Other tab, through the Cache Timeout parameter. (To bring up this window, right-click on the MSX's icon in RSM Console's map window, and choose Configure...)

Caching works two different ways, depending on an endpoint's registration mode (gatekeeper vs. OBP/mirror proxy). See *Subnet CAC* on page 83, the *Overview* section.

### **Keep-alive behavior**

The way registration keep-alive works varies according to whether the iServer acts as the registrar or a proxy to a separate registrar, such as a Broadsoft application server. These two behaviors are described below.

### **Locally-registered**

For endpoints that use the iServer as their registrar, the flow and results are as follows:

1. The endpoint sends to the iServer (the endpoint's registrar) a registration request (`REGISTER`) containing a timeout value. [example: 3600 seconds]
2. The iServer compares the value proposed in the registration request to one of the following:
  - 2.1 If the endpoint belongs to an iEdge group (*igrp*), the timeout value for that *igrp* [example: 60 seconds], **or**
  - 2.2 If the endpoint does not belong to an iEdge group (*none*), the iServer's system default parameter [example: 900 seconds]
3. If the timeout proposed by the endpoint in step 1 is equal to or shorter than the timeout obtained in step 2, minus one second [example 22.1:  $60 - 1 = 59$  seconds; or example 22.2,  $900 - 1 = 899$  seconds], then the value proposed by the endpoint in step 1 is used. Otherwise, the result of the above calculation [59 or 899 seconds in this example], is used.

In this example, the endpoint's proposed 3600-second timeout is longer than either 899 or 59 seconds, so either 59 seconds (if the endpoint's `igrp timeout = 60`) or 899 (if the endpoint is not a member of an *igrp*) is sent

back to the endpoint (via `expires`) as its required keep-alive registration message interval. The shorter interval is useful, for example, to keep firewall pinholes open between the MSX and the endpoint. Firewalls maintain their own timeouts, and this way the endpoint and MSX work together to ensure the pinhole stays open as long as needed.

### ***Registration by proxy***

When the MSX is acting in OBP mode or mirror proxy mode, the endpoint is actually registering with an external registrar server, not the MSX. In this case, the MSX forwards registration messages (both initial and keep-alive) to the external registrar, with optional throttling as described below.

### ***Keep-alive REGISTER forward throttling***

Ordinarily, when a SIP UA sends a REGISTER message to the MSX running in OBP mode, the MSX forwards that REGISTER to the designated registrar machine. In some cases, the rate at which a UA sends keep-alive REGISTERS can be higher than the registrar is able to receive and process them.

For such cases, an attribute named `obpxfactor` is available in `servercfg` to specify the frequency of REGISTERS forwarded to the registrar from the MSX. If, for example, a UA sends a REGISTER every 60 seconds, that rate could be reduced to once every 10 minutes, by specifying 600 seconds.

To set a value for this attribute, log into the iServer and enter:

```
nxconfig.pl -e obpxfactor -v value
```

where value is the number of seconds to set the `obpxfactor`.

For information on the timeout attribute, and how it works to set an endpoint's timeout interval, see *Inbound calls* on page 223.

If an endpoint fails to meet the iServer's shorter keep-alive interval, the iServer actively de-registers the endpoint from the remote registrar, provided the attribute that controls this feature is `servercfg`.

To enable this feature, log into the iServer and enter:

```
nxconfig.pl -e enable-autounregister -v 1
```

De-registration is accomplished by the MSX sending to the registrar a registration message for that endpoint with an `expires=0` parameter.

To apply this kind of throttling to an endpoint, make the endpoint a member of an `igrp`, and set that `igrp`'s parameter to the value you wish to impose on that endpoint. The endpoint will be required to keep itself alive at that rate, and the

registrar will receive keep-alive messages from the MSX at the less-frequent system timeout rate.

### ***Subnet-based throttling***

Transient endpoints that register with the iServer do so on the basis of the subnet from which their packets enter the network. In this case, the keep-alive interval is the one set for that subnet via its associated iEdge group (igrp).

*Subnet CAC* on page 83 includes the procedure for assigning an igrp based on an endpoint's subnet.

## **SIP outbound proxy and mirror proxy**

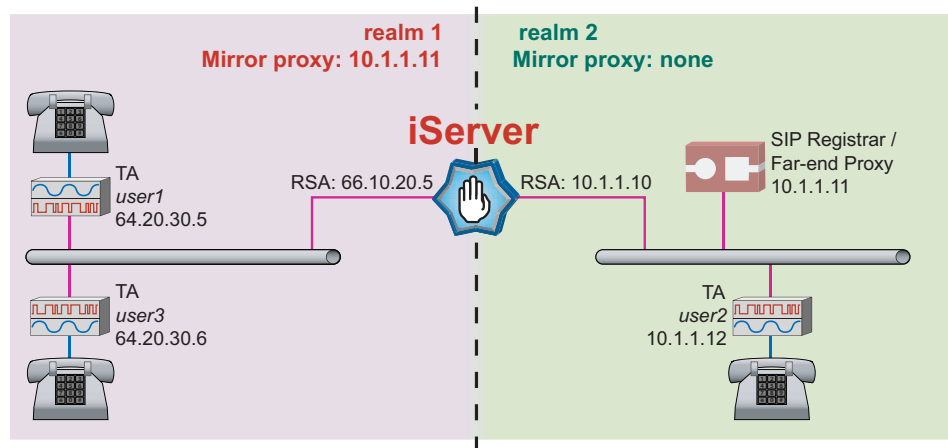
Proxy servers add a layer of indirection to network communications, and are often used to enhance security. They can also be used to hide network topologies when it's desirable to do so, in much the same way as media routing hides vendor and customer endpoints from the each other, thus preventing them from circumventing the VoIP transport provider. The MSX's *Outbound Proxy* (OBP) feature provides this functionality.

Normally, in order to use out-bound proxy (OBP) functionality, an endpoint must have built-in OBP support. For endpoints that do not support OBP intrinsically, MSX provides a "mirror proxy" feature.

*Mirror Proxy* is when the MSX acts as a proxy-for-the-proxy, so to speak. One specific realm on the MSX is configured as the mirror proxy realm. Endpoints without intrinsic OBP functionality register with this defined mirror proxy realm, and use it as their proxy by sending signaling messages to that realm's RSA.

The MSX, in turn, forwards the received request to the “far-end proxy/registrar”. For example, the machine actually providing proxy services to the endpoint, and with which it is registered. Figure 27 illustrates this.

**Figure 27. OBP / Mirror Proxy Topology**



### Implementation highlights

Configuration for Mirror Proxy use is based on the realm in which an endpoint resides. Realm configuration for mirror proxy is discussed in *Related CLI commands* on page 233.

SIP messages received from a realm configured in mirror proxy mode are forwarded to the “far-end proxy” server, after appropriate modifications to the header contents.

Calls from realms in mirror proxy mode can be made either to destinations inside the same realm, or in a different realm.

*Incoming* call flows are not affected by this feature.

### SIP Message flows

This section presents high-level descriptions of SIP message flows. The examples below are based on the network depicted in the diagram above.

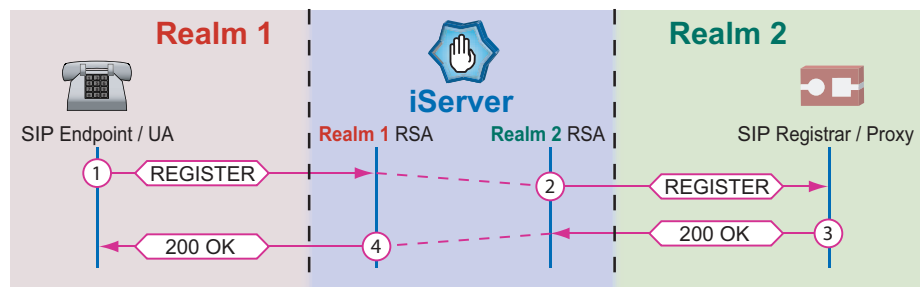
To make it easier to follow, the stand-alone proxy device in these scenarios, sometimes referred to as the “far-end proxy” is called the *proxy* in the following flow descriptions, and the MSX acting providing the “outbound proxy” service is merely referred to as the MSX.

Call setups still follow the basic INVITE, OK, ACK sequence, as noted.

### **Registration**

This flow is illustrated in Figure 28.

**Figure 28. Far-end Proxy Registration**



For user1's endpoint to register:

1. The endpoint sends a **REGISTER** message to the realm signaling address (RSA) for its realm, on the gatekeeper it is configured to use, which is the MSX.
2. The MSX forwards the **REGISTER** from the RSA for the realm in which the SIP Registrar/proxy is located to the proxy the endpoint will use. No originating-realm IP addresses are retained, thus effectively hiding the network topology.
3. The SIP Registrar/proxy replies **200 OK** to the RSA from which it received the **REGISTER**.
4. The MSX in turn replies **200 OK** to the registering endpoint, from the RSA for that realm. All IP addresses are mapped back to originating-realm addresses, again effectively hiding the network topology.

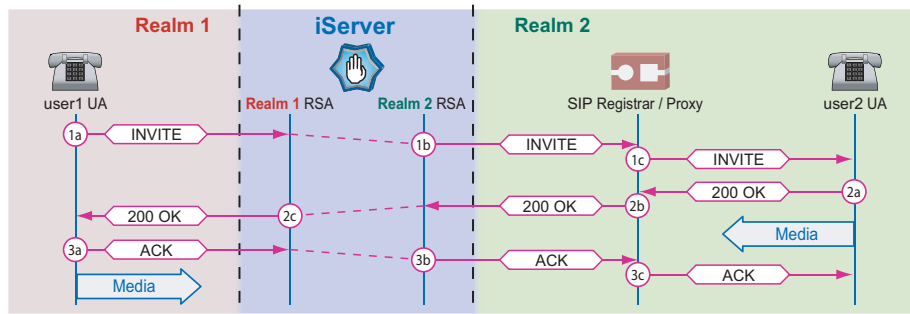
### **REGISTER caching**

Ordinarily, when a SIP UA sends a **REGISTER** message to the MSX running in OBP mode, the MSX unconditionally forwards that **REGISTER** to the designated registrar machine. In some cases, the rate at which a UA sends keep-alive **REGISTER** messages can be higher than the registrar machine is able to receive and process them, and in any event, places an unnecessary traffic burden on the network.

For such cases, the MSX provides the caching/throttling mechanism described in *Dynamic and transient registration* on page 224.

**Cross-realm call set-up**

In this scenario, a caller in one realm, user1, calls user2 in another realm (which is where the “far-end” proxy also resides). This flow is illustrated in Figure 29.

**Figure 29. Cross-realm OBP Call Setup**

Here are the steps:

**1. INVITE**

- 1a. user1 sends its INVITE to the RSA for realm 1, since that is the realm in which user1 resides.
- 1b. The MSX forwards the INVITE from the RSA for realm 2, to the proxy, where the destination endpoint also resides. All IP addresses from realm 1 are mapped to those for realm 2, where the destination endpoint and proxy reside, thus effectively hiding the transport network topology.
- 1c. The proxy then forwards the INVITE to the destination endpoint, which resides in the proxy's realm. user2's IP address is inserted into the INVITE, and a second `via` header is added with the proxy's IP address.

**2. 200 OK**

- 2a. user2 responds with a 200 OK sent back to the proxy from which it received the INVITE. (Note that user2 begins sending media packets any time after this.)
- 2b. The proxy receives the 200 OK, and relays it back to the RSA for the realm in which it's located on the MSX.

- 2c. The MSX relays the 200 OK back to the originating endpoint, user1. All IP addresses are mapped back to those for the realm in which user1 resides, which again hides the network topology.

### 3. ACK

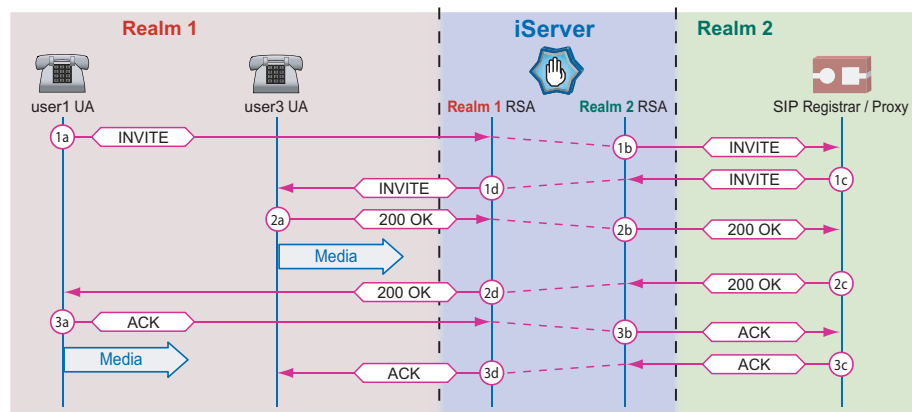
- 3a. user1 acknowledges the OK with an ACK sent back to the RSA for its realm.
- 3b. The MSX relays this ACK to the proxy from the realm 2 RSA (again, hiding the realm 1 addresses).
- 3c. The proxy relays this ACK to user2, and user1 begins sending media packets.

The conversation proceeds until either user1 or user2 sends a BYE.

#### In-realm call set-up

In this scenario, user1 calls user3. Both users reside in the same realm, but the “far-end” proxy resides in a different realm, so only the IP addresses used for proxy communication are in realm 2. Even though both user1 and user3 reside in the same realm, their locations are unknown to each other, and complete network topological anonymity is maintained, since each endpoint knows only of the MSX’s RSA, but neither the proxy’s nor the other endpoint’s IP addresses. This flow is illustrated in Figure 30.

**Figure 30. In-realm OBP Call Setup**



Here are the steps:

### 1. **INVITE**

- 1a. user1 sends the INVITE to the realm 1 RSA on the MSX.
- 1b. From the realm 2 RSA, the MSX forwards the INVITE to the proxy in realm 2. All IP addresses are mapped to ones appropriate for the realm 2 network. (All IP addresses are one of only two: the RSA for that realm, or the proxy's address.)
- 1c. The proxy "forwards" the INVITE to realm 2's RSA back on the MSX.
- 1d. The MSX forwards that INVITE to user3, with all IP addresses mapped back to those appropriate for realm 1 (user1's IP address appears nowhere in the message.)

### 2. **200 OK**

- 2a. user3 then responds to the INVITE by sending 200 OK back to its RSA on the MSX. (Note that user3 begins sending media packets any time after this.)
- 2b. The MSX forwards the 200 OK back to the proxy from its realm 2 RSA. Again, IP addresses are mapped back to those for realm 2.
- 2c. The proxy again "forwards" the 200 OK back to the MSX.
- 2d. The MSX forwards the 200 OK back to user1, which responds with ACK, as shown next.

### 3. **ACK**

- 3a. user1 acknowledges the OK by sending an ACK back to the RSA for its realm.
- 3b. The ACK is forwarded by the MSX to the proxy.
- 3c. The proxy "forwards" the ACK back to the MSX.
- 3d. The MSX forwards the ACK back to user3.

The conversation proceeds until either user1 or user3 sends a BYE.

### **Caveats**

- To enable the OBP feature use the `nxconfig.pl` utility to enable OBP and set the server type to `proxystateful`. See *Configuring global SIP parameters* on page 212 for a procedure.



- To use OBP, dynamic endpoint registration must be enabled. To enable it, log into the iServer and enter:

```
nxconfig.pl -e allow-dynamicendpoints -v 1
```

- The SIP registrar to be used must be configured as an MSX endpoint, and the SIP URI parameter for it must be set in RSM Console's SIP Protocol Parameters dialog.
- For OBP to work, FCE must also be enabled.

### **Related CLI commands**

As mentioned above, use of the out-bound proxy feature requires that the realms participating in OBP be configured for it. Specifically, the realm must be associated with the far-end proxy it will be using. For this, use the `cli realm edit` command, as follows:

```
cli realm edit realm name proxy_regid regid
cli realm edit realm name proxy_uport uport
```

The regid and uport you enter here must be those configured on the far-end proxy server that the realm named in realm name will use.

## **SIP-T support**

SIP (Session Initiation Protocol) is a protocol for internet session control. *SIP-T* specifies how to use SIP for Telephony. The MSX supports pass-through of SS7's ISDN "User Part" (abbreviated *ISUP*) in SIP messages, and forwarding the NANPA's ANI II digits.

The subsections below describe these two features, and provide a call-flow template.

### ***ISUP***

*ISUP* is sometimes used to define the protocol and procedures for setting up, managing, and releasing trunk circuits that carry voice and data calls over the PSTN. The basic process is that the gateway converts the SS7 *ISUP* in the incoming setup to SIP messages that carry the data through to the destination endpoint. Mid-call *ISUP* signaling messages are carried in SIP's INFO method.

Applicable RFCs include:

- 3372: SIP-T Context and Architecture
- 3204: MIME media types for ISUP and QSIG Objects

- 3578: ISUP Overlap Signaling to SIP (covers an obsolete signaling method still in use in some places in the world)

### **ANI II**

*ANI II digits* identify certain aspects of the call based on the type of facility from which the call was placed. For example, a pay phone in a prison, or a hotel or motel. SIP -T provides a field known as OLI (Originating Line Information), that carries the digits defined in ANI II<sup>2</sup>. When so configured, the MSX takes the OLI digits (sometimes also called *infodigits*) encoded into the SIP-T INVITE frame, and prepends them to the Calling Party Number field. If either SIP-T or OLI are not present, 00 is prepended instead.

To enable this capability, add an egress ANI route to the terminating endpoint's calling plan, with an ANI prefix of [v:cgp:cat] (including the square brackets). Figure 31 shows an example of such a route, which would then be bound to an egress gateway's calling plan.

---

2. The ANI II Digit definitions may be obtained from the NANPA's Web site: <http://www.nanpa.com/>.

**Figure 31. ANI II Digit-Forwarding Route**

**Add Calling Plan Route**

Add new route

Group: admin

Route Name: Prepend-OLI

☒ ANI type ☐ DNIS type

**ANI**

Calling Party #:  Length:  ☒ Length unspecified

Prefix: [vcgpcat] Usage [help](#)

**DNIS**

Called Party #:  Length:  ☐ Length unspecified

Prefix:  ☐ No Digit Strip

☐ Default Route

☐ Reject

☐ Sticky

**Properties**

Applied at: ☐ Ingress ☒ Egress ☐ Transit

Template: ☐

Add Clear Close

### Call Flows

A sample of a SIP-T call flow is below.

**Call Set-up**

PSTN	MGC#1	iSrvr	MGC#2	PSTN
---IAM--->				
	----INVITE-->			
	(IAM)			
	<-100 TRYING-			
		----INVITE-->	----IAM----->	
		(IAM)		
			<----ACM-----	
		<----18x-----		
	<----18x-----	(ACM)		
<--ACM-----	(ACM)			
			<----ANM-----	
		<--200 OK----		
	<--200 OK----	(ANM)		
<--ANM-----	(ANM)			
	----ACK----->			
		----ACK----->		
=====Conversation=====				

**Call Release**

PSTN	MGC#1	iSrvr	MGC#2	PSTN
=====Conversation=====				
---REL--->				
<--RLC-----	---- BYE----->			
	(REL)	----BYE----->		
		(REL)		
	<----200 OK---		----REL----->	
		<---200 OK---		
			<----RLC-----	

**Caveats**

Issues to be aware of when implementing this ISUP pass-through include:

- ISUP/QSIG packets from SIP are not carried through IWF, only SIP-to-SIP. MSX's pure H.323 support does not include ISUP either.

- If a destination endpoint does not support multipart/mime message bodies, and therefore responds with *415 Unsupported Media Type*, the MSX passes the failure back to the originating endpoint.

## SIP privacy

The MSX provides limited support for *SIP Privacy*, which affects caller identification information content and display.

IETF RFC 3325, and a separate, prior proposed standard also in use, called *draft-ietf-sip-privacy-01.txt*, each define a set of SIP extensions that enable a network of trusted SIP servers to authoritatively assert the identity of authenticated users, and to control the forwarding of caller-identifying information. A subset of the full specifications is supported. This section details the MSX's support for those two standards.

### ***Identity assertion via trusted servers***

A caller's identity can be asserted by a *trusted* server. A trusted server is defined as one explicitly known to the MSX because of being registered with it, or registered with the same Sgatekeeper.

### ***Untrusted endpoints***

Endpoints can be designated as *untrusted* (*trusted* being the default). If an endpoint is designated as *untrusted*, private information is suppressed in INVITEs sent to it, as follows:

- The privacy headers are stripped out of the request entirely
- The `From:` header is altered to read `Anonymous`, according to the SIP Privacy approach used, as detailed in the sections *RFC 3325 support* and *Support for draft-ietf-sip-privacy-01.txt* that follow.

### **Configuring endpoint trust using RSM console**

1. In the tree, right-click the endpoint that you want to configure, then select **Modify** from the pop-up menu that opens.  
The **Provision an Endpoint** window opens.
2. If it is not already selected, select the **Protocol** tab.

3. Select the SIP check-box (highlighted in Figure 32), then click Configure.

**Figure 32. SIP endpoint configuration access**

The screenshot shows the 'Provision an Endpoint' dialog box with the 'Protocol' tab selected. The dialog has several sections:

- Protocol Settings:** Contains a 'Gateway/Proxy' checkbox, a 'Priority' text field, and an 'LNP' dropdown menu currently set to '<none>'.
- SIP/H323:** Contains two checkboxes, 'SIP' and 'H.323'. The 'SIP' checkbox is highlighted with a grey circle. Next to each checkbox is a 'Configure' button. Below these is a 'URI (Sip / H323)' text field.
- Trunk Group:** Contains four text fields: 'Src. Trunk Group:', 'Dest. Trunk Group:', 'New Src. Ingress Trunk Group:', and 'New Src. Egress Trunk Group:'. Below these are two checkboxes: 'Send Dest. Trunk Group' and 'Remove Src. Trunk Group'.

At the bottom of the dialog are 'OK' and 'Cancel' buttons.

The **SIP Protocol Parameters** window opens.

4. If you want to change the trust setting, click to select or clear the SIP host untrusted checkbox.

**Figure 33. SIP host trust option**

5. Click OK.

The system implements your endpoint trust selection and the **SIP Protocol Parameters** window closes.

### **Configuring endpoint trust via CLI**

The following CLI command sets endpoint trust:

```
cli iedge edit regid uport trust [disable | enable]
```

Remember that endpoints are trusted by default. To designate an endpoint as untrusted, issue this command with a `trust disable` option.

### ***RFC 3325 support***

The MSX's support of RFC 3325 features is detailed in this section.

According to the RFC, the extensions it defines:

...enable a network of trusted SIP servers to assert the identity of end users or end systems, and to convey indications of end-user requested privacy.

The subsections below describe the specifics of this RFC that apply to MSX 3.1.

**Identity assertion**

RFC 3325 (paragraph 9.1) adds a new SIP header field, P-Asserted-Identity, which contains the “trusted” identity of the caller. For example

```
P-Asserted-Identity: "Alice" <sip:alice@nextone.com>
```

When this header is given, the caller’s identity (i.e., the one sent to the called party) becomes the one given in it.

**Privacy level control**

The specifications in the RFC (paragraph 9.3) allow callers to set various levels of privacy when placing calls to *untrusted* entities. Trusted endpoints receive all the identity information, regardless of these settings. They can suppress their entire identity, just their URI (sip:alice@nextone.com), or just the name portion of the complete ID ("alice"). This is done with the Privacy header, as shown in Table 22.

**Table 22. Privacy Header Options**

Option	Description
id	Suppresses the entire ID line. How the Caller ID display depends on exactly how the receiving end handles this header.
name	Suppresses just the “name” portion of the caller’s identity. In the example above, this means only sip:alice@nextone.com is sent to the caller’s UA, not "Alice".
URI	Suppresses just the URI portion of the caller’s identity. In the example above, this means only "Alice" is sent to the caller’s UA, not sip:alice@nextone.com.

The header line looks like the following example:

```
Privacy:id
```

**RFC 3325 example 1**

The following example illustrates an RFC 3325 scenario where the caller asserts their P-Asserted-Identity as a replacement for the one provided in the From header, and requests blocking of the complete caller-id identity from untrusted endpoints.

```
INVITE sip:0333610123@nextone.com:5060;user=phone SIP/2.0
Via:SIP/2.0/UDP 210.131.74.145;branch=z9hG4bK
From:<sip:anonymous@anonymous.invalid>;tag=1384820071-
1070071993639
To:<sip:0333610123@nextone.com>
Call-ID:111313063929
```



```

CSeq:312568468 INVITE
Contact:<sip:210.131.74.145>
P-Asserted-Identity:"Alice"<sip:alice@nextone.com>
Privacy:id
Min-SE:180
Content-Type:application/sdp
Max-Forwards:10
Content-Length:153
----SDP----

```

### **RFC 3325 example 2**

This example illustrates an RFC 3325 scenario where the caller asserts their P-Asserted-Identity as a replacement for the one provided in the From header, to be sent to all recipients, trusted or untrusted.

```

INVITE sip:0333610123@nextone.com:5060;user=phone SIP/2.0
Via:SIP/2.0/UDP 210.131.74.145;branch=z9hG4bK
From:<sip:anonymous@anonymous.invalid>;tag=1384820071-
1070071993639
To:<sip:0333610123@nextone.com>
Call-ID:111313063929
CSeq:312568468 INVITE
Contact:<sip:210.131.74.145>
P-Asserted-Identity:"Alice"<sip:alice@nextone.com>
Min-SE:180
Content-Type:application/sdp
Max-Forwards:10
Content-Length:153
----SDP----

```

### **Support for draft-ietf-sip-privacy-01.txt**

This draft standard approaches the issue of privacy and identity assertion in a different way. It is also supported, for backward compatibility.

#### **Identity assertion**

Assertion of identity in this “draft” standard uses different headers than the RFC 3325 method uses. With the draft, identity is asserted with the Remote-Party-ID header. For example:

```

Remote-Party-ID:
"Alice"<sip:alice@nextone.com>;screen=yes;party=calling;privacy=full

```

The draft standard details the parameters listed in this example.

#### **Privacy level control**

The level of privacy is controlled by the parameters in the Remote-Party-ID header, and affect what information is sent to untrusted entities. Possible values and their definitions are shown in Table 23.

**Table 23. ID Parameter Options**

Parameter	Option	Description
screen	no	(Default) Indicates the Remote-Party-ID was either not verified successfully by the proxy or the proxy received the message from an untrusted entity.
	yes	Indicates the Remote-Party-ID was verified successfully by the proxy itself or the proxy received the message from a trusted proxy with this indication.
party	calling	(Default for INVITE) The information in this header is for the calling party.
	called	(Default for response) The information in this header is for the called party.
privacy	full	Suppresses the entire ID line. How the Caller ID display depends on exactly how the receiving end handles this header.
	name	Suppresses just the "name" portion of the caller's identity. In the example above, this means only sip:alice@nextone.com is sent to the caller's UA, not "Alice".
	uri	Suppresses just the address specification portion of the caller's identity. In the example above, this means only "Alice" is sent to the caller's UA, not sip:alice@nextone.com.
	off	Privacy functions are explicitly <i>disabled</i> for untrusted entities (i.e., privacy is turned off).

**Draft Spec example 1**

In this example, the originating UA sends an INVITE requesting caller ID blocking. Note the inclusion of a Proxy-Require header following the Remote-Party-ID, which is a draft spec requirement for INVITEs using these extensions.

```

INVITE sip:0333610123@nextone.com:5060;user=phone SIP/2.0
Via:SIP/2.0/UDP 210.131.74.145;branch=z9hG4bK
From:<sip:anonymous@localhost>;tag=1384820071-1070071993639
To:<sip:0333610123@nextone.com>
Call-ID:111313063929
CSeq:312568468 INVITE
Contact:<sip:210.131.74.145>
Remote-Party-
ID: "Alice"<sip:alice@nextone.com>;screen=yes;privacy=full

```

```

Proxy-Require: privacy
Min-SE:180
Content-Type:application/sdp
Max-Forwards:10
Content-Length:153
-----SDP-----

```

### **Draft Spec example 2**

In this example, the originating UA sends an INVITE with privacy extensions turned off. The remote party ID is sent to all SIP entities, whether trusted or untrusted. Note the absence of a Proxy-Require header, which is not required when privacy is turned off.

```

INVITE sip:0333610123@nextone.com:5060;user=phone SIP/2.0
Via:SIP/2.0/UDP 210.131.74.145;branch=z9hG4bK
From:<sip:anonymous@anonymous.invalid>;tag=1384820071-1070071993639
To:<sip:0333610123@nextone.com>
Call-ID:111313063929
CSeq:312568468 INVITE
Contact:<sip:210.131.74.145>
Remote-Party-ID:"Alice"<sip:alice@nextone.com>;screen=yes;privacy=off
Min-SE:180
Content-Type:application/sdp
Max-Forwards:10
Content-Length:153
-----SDP-----

```

### **Caveats**

Please note that the draft specification states the following points (among others):

- There must not be more than one party parameter in a Remote-Party-ID. The defaults apply only when *no* party parameter is specified.
- UACs<sup>3</sup> that wish to use the extensions defined {in the draft specification} MUST include a Proxy-Require header in the initial INVITE request containing the option tag privacy.
- When such a UAC initiates a call, it SHOULD include a calling subscriber Remote-Party-ID header field in the initial INVITE request in order to identify the originator of the call.
- This Remote-Party-ID MUST contain a SIP-URL identifying the UAC and MAY contain a display-name for the UAC as well.

---

3. "User Agent Clients". That is, the UA program that runs on the client device.

- The party type SHOULD be set to `calling` and the identity type SHOULD be set to `subscriber`, however other types of party and identity information may be included as well.
- If `Remote-Party-ID` privacy is desired, the UAC MUST include a privacy token set to one or more of `uri`, `name` or `full`.

### Configuring Privacy

Support for the two supported methods of privacy is via CLI command, on a per-endpoint basis. The new command's syntax is:

```
cli iedge edit regid [low [-high]] privacy value
```

...where value is one of the following three strings:

```
rfc3325
draft01 ("0" = zero)
both (default)
```

Note that as the default, `both` is automatically assigned to a SIP endpoint upon initial configuration. `both` means that the endpoint supports both RFC 3325 and the draft specification.

### MSX behavior

The MSX treats messages containing privacy information according to the following rules:

- Detects Privacy headers in an incoming INVITE
- Determines destination endpoint and its privacy capabilities
- If the destination endpoint supports the incoming privacy type, the privacy headers are passed through to the destination
- If the destination endpoint does not support the RFC3325 specification, the MSX translates from RFC3325 messages to those conforming to the draft specification.

Normally, when the MSX receives an INVITE with an ISUP portion, it extracts the address information from the Charge Number parameter in the encapsulated ISUP message and uses the digits in the Charge Number in the outgoing INVITEs and SETUPs as the name in the "From" header and Calling Party number respectively. However, if the incoming SIP INVITE contains either the P-Asserted Identity header or Remote-Party Identity and the destination endpoint's trust is `ENABLED`, then the name portion of the "From" header in the outgoing INVITE from the iServer will contain the address information of the Charge Number and also the privacy headers.

**Privacy behavior**

Tables 24 through 42 detail the MSX's SIP Privacy behaviors across protocols (SIP and H.323) and privacy standards (Draft-01 and RFC 3325), and with or without Caller ID Blocking enabled.

**Table 24. SIP Origination (no privacy) to SIP Destination**

INPUT			OUTPUT		
Incoming Privacy	Destination trust	Caller ID blocking	From header	Display name in From header	Privacy headers
No Privacy	Trusted	Disabled	sip:111@10.2.0.93	Nextone	None
No Privacy	Untrusted	Disabled	sip:111@10.2.0.93	Nextone	None
No Privacy	Trusted (Dest is Draft 01)	Enabled	sip:anonymous@localhost	Anonymous	RPID: "Nextone" <sip:sipp@10.2.0.93>; privacy=full
No Privacy	Untrusted (Dest is Draft 01)	Enabled	sip:anonymous@localhost	Anonymous	None
No Privacy	Trusted (Dest is RFC 3325)	Enabled	sip:anonymous@anonymous.invalid	Anonymous	PAID: "Nextone" <sip:sipp@10.2.0.93> Privacy: id
No Privacy	Untrusted (Dest is RFC 3325)	Enabled	sip:anonymous@anonymous.invalid	Anonymous	None
No Privacy	Trusted (Dest is Both)	Enabled	sip:anonymous@anonymous.invalid	Anonymous	PAID: "Nextone" <sip:sipp@10.2.0.93> Privacy: id
No Privacy	Untrusted (Dest is Both)	Enabled	sip:anonymous@anonymous.invalid	Anonymous	None

**Table 25. SIP (RFC 3325) Origination to SIP (Both) Destination,  
Caller ID Blocking Disabled**

INPUT			OUTPUT		
Incoming Privacy	Destination trust	Caller ID blocking	From header	Display name in From header	Privacy headers
From: "Nextone" <sip:anonymous@anonymous.invalid>  PAID: <sip:sipp@172.16.1.62> Privacy: id	Trusted	Disabled	sip:sipp@10.2.0.93	Nextone	PAID: <sip:sipp@172.16.1.62> Privacy: id
From: "Nextone" <sip:anonymous@anonymous.invalid>  PAID: <sip:sipp@172.16.1.62>	Trusted	Disabled	sip:sipp@10.2.0.93	Nextone	PAID: <sip:sipp@172.16.1.62>
From: "Nextone" <sip:anonymous@anonymous.invalid>  PAID: <sip:sipp@172.16.1.62> Privacy: id	Untrusted	Disabled	sip:anonymous@anonymous.invalid	Anonymous	None
From: "Nextone" <sip:anonymous@anonymous.invalid>  PAID: <sip:sipp@172.16.1.62>	Untrusted	Disabled	sip:anonymous@10.2.0.93	Nextone	None

**Table 26. SIP (RFC 3325) Origination to SIP (Both/ RFC 3325) Destination,  
Caller ID Blocking Enabled**

INPUT			OUTPUT		
Incoming Privacy	Destination Trust	Caller ID Blocking	From Header	Display Name in From Header	Privacy headers
From: "Nextone" <sip:anonymous@anonymous.invalid>  PAID: <sip:sipp@172.16.1.62> Privacy: id	Trusted	Enabled	sip:anonymous@anonymous.invalid	Anonymous	PAID: <sip:sipp@172.16.1.62> Privacy: id
From: "Nextone" <sip:anonymous@anonymous.invalid>  PAID: <sip:sipp@172.16.1.62>	Trusted	Enabled	sip:anonymous@anonymous.invalid	Anonymous	PAID: <sip:sipp@172.16.1.62>
From: "Nextone" <sip:anonymous@anonymous.invalid>  PAID: <sip:sipp@172.16.1.62> Privacy: id	Untrusted	Enabled	sip:anonymous@anonymous.invalid	Anonymous	None
From: "Nextone" <sip:anonymous@anonymous.invalid>  PAID: <sip:sipp@172.16.1.62>	Untrusted	Enabled	sip:anonymous@anonymous.invalid	Anonymous	None

**Table 27. SIP (RFC 3325) Origination to SIP (Draft 01) Destination,  
Caller ID Blocking Disabled**

INPUT			OUTPUT		
Incoming Privacy	Destination trust	Caller ID blocking	From header	Display name in From header	Privacy headers
From: "Nextone" <sip:anonymous@anonymous.invalid>  PAID: <sip:sipp@172.16.1.62> Privacy: id	Trusted	Disabled	sip:anonymous@localhost	Anonymous	RPID: <sip:sipp@172.16.1.62>; Privacy= full  Proxy-Require: Privacy
From: "Nextone" <sip:anonymous@anonymous.invalid>  PAID: <sip:sipp@172.16.1.62>	Trusted	Disabled	sip:sipp@10.2.0.93	Blank display-name	RPID: <sip:sipp@172.16.1.62>; Privacy= off  Proxy-Require: Privacy
From: "Nextone" <sip:anonymous@anonymous.invalid>  PAID: <sip:sipp@172.16.1.62> Privacy: id	Untrusted	Disabled	sip:anonymous@localhost	Anonymous	None
From: "Nextone" <sip:anonymous@anonymous.invalid>  PAID: <sip:sipp@172.16.1.62>	Untrusted	Disabled	sip:sipp@10.2.0.93	Blank display-name	None



**Table 28. SIP (RFC 3325) Origination to SIP (Draft 01) Destination,  
Caller ID Blocking Enabled**

INPUT			OUTPUT		
Incoming Privacy	Destination trust	Caller ID blocking	From header	Display name in From header	Privacy headers
From: "Nextone" <sip:anonymous@anonymous.invalid>  PAID: <sip:sipp@172.16.1.62> Privacy: id	Trusted	Enabled	sip:anonymous@localhost	Anonymous	RPID: <sip:sipp@172.16.1.62>; Privacy= full  Proxy-Require: Privacy
From: "Nextone" <sip:anonymous@anonymous.invalid>  PAID: <sip:sipp@172.16.1.62>	Trusted	Enabled	sip:anonymous@localhost	Anonymous	RPID: <sip:sipp@172.16.1.62>; Privacy= off  Proxy-Require: Privacy
From: "Nextone" <sip:anonymous@anonymous.invalid>  PAID: <sip:sipp@172.16.1.62> Privacy: id	Untrusted	Enabled	sip:anonymous@localhost	Anonymous	None
From: "Nextone" <sip:anonymous@anonymous.invalid>  PAID: <sip:sipp@172.16.1.62>	Untrusted	Enabled	sip:anonymous@localhost	Anonymous	None

**Table 29. SIP (Draft 01) Origination to SIP (Draft 01) Destination,  
Caller ID Blocking Disabled**

INPUT			OUTPUT		
Incoming Privacy	Destination trust	Caller ID blocking	From header	Display name in From header	Privacy headers
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62>; screen=no; party=calling; privacy=full  Proxy-Require: Privacy	Trusted	Disabled	sip:sipp@10.2.0.93	Nextone	RPID: <sip:sipp@172.16.1.62>; screen=no; party=calling; Privacy= full  Proxy-Require: Privacy
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62>; screen=no; party=calling; privacy=off  Proxy-Require: Privacy	Trusted	Disabled	sip:sipp@10.2.0.93	Nextone	RPID: <sip:sipp@172.16.1.62>; screen=no; party=calling; Privacy= of  Proxy-Require: Privacy
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62>; screen=no; party=calling; privacy=full  Proxy-Require: Privacy	Untrusted	Disabled	sip:anonymous@localhost	Anonymous	None
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62>; screen=no; party=calling; privacy=off  Proxy-Require: Privacy	Untrusted	Disabled	sip:anonymous@10.2.0.93	Nextone	None

**Table 30. SIP (Draft 01) Origination to SIP (Draft 01) Destination,  
Caller ID Blocking Enabled**

INPUT			OUTPUT		
Incoming Privacy	Destination trust	Caller ID blocking	From header	Display name in From header	Privacy headers
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62>; screen=no; party=calling; privacy=full  Proxy-Require: Privacy	Trusted	Enabled	sip:anonymous@localhost	Nextone	RPID: <sip:sipp@172.16.1.62>; screen=no; party=calling; Privacy= full  Proxy-Require: Privacy
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62>; screen=no; party=calling; privacy=off  Proxy-Require: Privacy	Trusted	Enabled	sip:anonymous@localhost	Nextone	RPID: <sip:sipp@172.16.1.62>;screen=no ;party=calling; Privacy= off  Proxy-Require: Privacy
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62>; screen=no; party=calling; privacy=full  Proxy-Require: Privacy	Untrusted	Enabled	sip:anonymous@localhost	Anonymous	None
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62>; screen=no; party=calling; privacy=off  Proxy-Require: Privacy	Untrusted	Enabled	sip:anonymous@localhost	Nextone	None

**Table 31. SIP (Draft 01) Origination to SIP (RFC 3325) Destination,  
Caller ID Blocking Disabled  
(Draft 01 to RFC 3325 conversion not supported)**

INPUT			OUTPUT		
Incoming Privacy	Destination trust	Caller ID blocking	From header	Display name in From header	Privacy headers
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62>; screen=no; party=calling; privacy=full  Proxy-Require: Privacy	Trusted	Disabled	sip:anonymous@10.2.0.93	Nextone	RPID: <sip:sipp@172.16.1.62>; screen=no; party=calling; Privacy= full  Proxy-Require: Privacy
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62>; screen=no; party=calling; privacy=off  Proxy-Require: Privacy	Trusted	Disabled	sip:anonymous@10.2.0.93	Nextone	RPID: <sip:sipp@172.16.1.62>; screen=no; party=calling; Privacy= off  Proxy-Require: Privacy
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62>; screen=no; party=calling; privacy=full  Proxy-Require: Privacy	Untrusted	Disabled	sip:anonymous@anonymous.invalid	Anonymous	None
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62>; screen=no; party=calling; privacy=off  Proxy-Require: Privacy	Untrusted	Disabled	sip:anonymous@10.2.0.93	Nextone	None

**Table 32. SIP (Draft 01) Origination to SIP (RFC 3325) Destination,  
Caller ID Blocking Enabled  
(Draft 01 to RFC 3325 conversion not supported)**

INPUT			OUTPUT		
Incoming Privacy	Destination trust	Caller ID blocking	From header	Display name in From header	Privacy headers
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62> ;screen=no; party=calling; privacy=full  Proxy-Require: Privacy	Trusted	Enabled	sip:anonymous@anonymous.invalid	Anonymous	RPID: <sip:sipp@172.16.1.62>; screen=no; party=calling; Privacy= full  Proxy-Require: Privacy
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62>; screen=no; party=calling; privacy=off  Proxy-Require: Privacy	Trusted	Enabled	sip:anonymous@anonymous.invalid	Anonymous	RPID: <sip:sipp@172.16.1.62>; screen=no; party=calling; Privacy= off  Proxy-Require: Privacy
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62>; screen=no; party=calling; privacy=full  Proxy-Require: Privacy	Untrusted	Enabled	sip:anonymous@anonymous.invalid	Anonymous	None
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62>; screen=no; party=calling; privacy=off  Proxy-Require: Privacy	Untrusted	Enabled	sip:anonymous@anonymous.invalid	Anonymous	None

**Table 33. H.323 to H.323, Trust enabled**

INPUT			OUTPUT		
Incoming Privacy	Destination trust	Q.931 display	Calling Party Number (CPN)	Display IE	Privacy indicators
No Privacy	Trusted	Enabled	3333	Nextone	None
No Privacy	Trusted	Disabled	3333	None	None
Presentation Indicator = Allowed (0)/ Unavailable (2) Screening Indicator = Not Screened (0)	Trusted	Enabled	3333	Nextone	PI = 0/2; SI = 0
Presentation Indicator = Allowed (0)/ Unavailable (2) Screening Indicator = Not Screened (0)	Trusted	Disabled	3333	None	PI = 0/2; SI = 0
Presentation Indicator = Allowed (0)/ Unavailable (2) Screening Indicator = Verified Pass / Verified Fail / Network	Trusted	Enabled	3333	Nextone	PI = 0/2; SI = 1/2/3
Presentation Indicator = Allowed (0)/ Unavailable (2) Screening Indicator = Verified Pass / Verified Fail / Network	Trusted	Disabled	3333	None	PI = 0/2; SI = 1/2/3
Presentation Indicator = Restricted (1) Screening Indicator = Not Screened (0)	Trusted	Enabled	3333	Nextone	PI = 1; SI = 0
Presentation Indicator = Restricted (1) Screening Indicator = Not Screened (0)	Trusted	Disabled	3333	None	PI = 1; SI = 0
Presentation Indicator = Restricted (1) Screening Indicator = Verified Pass / Verified Fail / Network	Trusted	Enabled	3333	Nextone	PI = 1; SI = 1/2/3

**Table 34. H.323 to H.323, Trust disabled**

INPUT			OUTPUT		
Incoming Privacy	Destination trust	Q.931 display	Calling Party Number (CPN)	Display IE	Privacy indicators
No Privacy	UnTrusted	Enabled	3333	Nextone	None
No Privacy	UnTrusted	Disabled	3333	None	None
Presentation Indicator = Allowed (0)/ Unavailable (2) Screening Indicator = Not Screened (0)	UnTrusted	Enabled	3333	Nextone	PI = 0/2; SI = 0
Presentation Indicator = Allowed (0)/ Unavailable (2) Screening Indicator = Not Screened (0)	UnTrusted	Disabled	3333	None	PI = 0/2; SI = 0
Presentation Indicator = Allowed (0)/ Unavailable (2) Screening Indicator = Verified Pass / Verified Fail / Network	UnTrusted	Enabled	Anonymous	Nextone	PI = 0/2; SI = 1/2/3
Presentation Indicator = Allowed (0)/ Unavailable (2) Screening Indicator = Verified Pass / Verified Fail / Network	UnTrusted	Disabled	Anonymous	None	PI = 0/2; SI = 1/2/3
Presentation Indicator = Restricted (1) Screening Indicator = Not Screened (0)	UnTrusted	Enabled	Anonymous	Nextone	PI = 1; SI = 0
Presentation Indicator = Restricted (1) Screening Indicator = Not Screened (0)	UnTrusted	Disabled	Anonymous	None	PI = 1; SI = 0
Presentation Indicator = Restricted (1) Screening Indicator = Verified Pass / Verified Fail / Network	UnTrusted	Enabled	Anonymous	Nextone	PI = 1; SI = 1/2/3

**Table 35. H.323 to SIP (RFC 3325), Trust enabled**

INPUT			OUTPUT		
Incoming Privacy	Destination trust	Q.931 display	From header	Display name in From header	Privacy indicators
No Privacy	Trusted	NA	sip:3333@10.2.0.93:5060	Nextone	None
Presentation Indicator = Allowed (0)/ Unavailable (2) Screening Indicator = Not Screened (0)	Trusted	NA	sip:3333@10.2.0.93:5060	Nextone	None
Presentation Indicator = Allowed (0)/ Unavailable (2) Screening Indicator = Verified Pass / Verified Fail / Network	Trusted	NA	sip:3333@10.2.0.93:5060	Nextone	PAID: <sip:3333@10.2.0.93> Privacy: id
Presentation Indicator = Restricted (1) Screening Indicator = Not Screened (0)	Trusted	NA	Anonymous	Nextone	PAID: <sip:3333@10.2.0.93> Privacy: id
Presentation Indicator = Restricted (1) Screening Indicator = Verified Pass / Verified Fail / Network	Trusted	NA	Anonymous	Nextone	PAID: <sip:3333@10.2.0.93> Privacy: id

**Table 36. H.323 to SIP (RFC 3325), Trust disabled**

INPUT			OUTPUT		
Incoming Privacy	Destination trust	Q.931 display	From header	Display name in From header	Privacy indicators
No Privacy	Untrusted	NA	sip:3333@10.2.0.93:5060	Nextone	None
Presentation Indicator = Allowed (0)/ Unavailable (2) Screening Indicator = Not Screened (0)	Untrusted	NA	sip:3333@10.2.0.93:5060	Nextone	None
Presentation Indicator = Allowed (0)/ Unavailable (2) Screening Indicator = Verified Pass / Verified Fail / Network	Untrusted	NA	sip:anonymous@anonymous.invalid	Anonymous	None
Presentation Indicator = Restricted (1) Screening Indicator = Not Screened (0)	Untrusted	NA	sip:anonymous@anonymous.invalid	Anonymous	None
Presentation Indicator = Restricted (1) Screening Indicator = Verified Pass / Verified Fail / Network	Untrusted	NA	sip:anonymous@anonymous.invalid	Anonymous	None



**Table 37. H.323 to SIP (Draft 01), Trust enabled**

INPUT			OUTPUT		
Incoming Privacy	Destination trust	Q.931 display	From header	Display name in From header	Privacy indicators
No Privacy	Trusted	NA	sip:3333@10.2.0.93:5060	Nextone	None
Presentation Indicator = Allowed (0)/ Unavailable (2) Screening Indicator = Not Screened (0)	Trusted	NA	sip:3333@10.2.0.93:5060	Nextone	None
Presentation Indicator = Allowed (0)/ Unavailable (2) Screening Indicator = Verified Pass / Verified Fail / Network	Trusted	NA	sip:anonymous@localhost	Anonymous	RPID: <sip:3333@10.2.0.93>; Privacy: full Proxy-Require: Privacy
Presentation Indicator = Restricted (1) Screening Indicator = Not Screened (0)	Trusted	NA	sip:anonymous@localhost	Anonymous	RPID: <sip:3333@10.2.0.93>; Privacy: full Proxy-Require: Privacy
Presentation Indicator = Restricted (1) Screening Indicator = Verified Pass / Verified Fail / Network	Trusted	NA	sip:anonymous@localhost	Anonymous	RPID: <sip:3333@10.2.0.93>; Privacy: full Proxy-Require: Privacy

**Table 38. H.323 to SIP (Draft 01), Trust disabled**

INPUT			OUTPUT		
Incoming Privacy	Destination trust	Q.931 display	From header	Display name in From header	Privacy indicators
No Privacy	Untrusted	NA	sip:3333@10.2.0.93:5060	Nextone	None
Presentation Indicator = Allowed (0)/ Unavailable (2) Screening Indicator = Not Screened (0)	Untrusted	NA	sip:3333@10.2.0.93:5060	Nextone	None
Presentation Indicator = Allowed (0)/ Unavailable (2) Screening Indicator = Verified Pass / Verified Fail / Network	Untrusted	NA	sip:anonymous@localhost	Anonymous	None
Presentation Indicator = Restricted (1) Screening Indicator = Not Screened (0)	Untrusted	NA	sip:anonymous@localhost	Anonymous	None
Presentation Indicator = Restricted (1) Screening Indicator = Verified Pass / Verified Fail / Network	Untrusted	NA	sip:anonymous@localhost	Anonymous	None

**Table 39. SIP (RFC 3325) to H.323, Caller ID blocking disabled**

INPUT			OUTPUT		
Incoming Privacy	Destination trust	Q.931 display	Calling Party Number (CPN)	Display IE	Privacy indicators
From: "Nextone" <sip:anonymous@anonymous.invalid> PAID: <sip:sipp@172.16.1.62> Privacy: id	Trusted	Disabled	No CPN	None	H.225 PI = Restricted (1)
From: "Nextone" <sip:anonymous@anonymous.invalid> PAID: <sip:sipp@172.16.1.62> Privacy: id	Trusted	Enabled	No CPN	Nextone	H.225 PI = Restricted (1)
From: "Nextone" <sip:anonymous@anonymous.invalid> PAID: <sip:sipp@172.16.1.62> Privacy: id	Untrusted	Disabled	No CPN	None	H.225 PI = Restricted (1)
From: "Nextone" <sip:anonymous@anonymous.invalid> PAID: <sip:sipp@172.16.1.62> Privacy: id	Untrusted	Enabled	No CPN	Nextone	H.225 PI = Restricted (1)

**Table 40. SIP (RFC 3325) to H.323, Caller ID blocking enabled**

INPUT			OUTPUT		
Incoming Privacy	Destination trust	Q.931 display	Calling Party Number (CPN)	Display IE	Privacy indicators
From: "Nextone" <sip:anonymous@anonymous.invalid> PAID: <sip:sipp@172.16.1.62> Privacy: id	Trusted	Disable	No CPN	None	H.225 PI = Restricted (1)
From: "Nextone" <sip:anonymous@anonymous.invalid> PAID: <sip:sipp@172.16.1.62> Privacy: id	Trusted	Enable	No CPN	Anonymous	H.225 PI = Restricted (1)
From: "Nextone" <sip:anonymous@anonymous.invalid> PAID: <sip:sipp@172.16.1.62> Privacy: id	Untrusted	Disable	No CPN	None	H.225 PI = Restricted (1)
From: "Nextone" <sip:anonymous@anonymous.invalid> PAID: <sip:sipp@172.16.1.62> Privacy: id	Untrusted	Enable	No CPN	Anonymous	H.225 PI = Restricted (1)

**Table 41. SIP (Draft 01) to H.323, Caller ID blocking disabled**

INPUT			OUTPUT		
Incoming Privacy	Destination trust	Q.931 display	Calling Party Number (CPN)	Display IE	Privacy indicators
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62>; screen=no; party=calling; privacy=off  Proxy-Require: Privacy	Trusted	Disable	No CPN	None	H.225 PI = Restricted (1), SI = Not Screened (0)
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62>; screen=no; party=calling; privacy=off  Proxy-Require: Privacy	Trusted	Enable	No CPN	Nextone	H.225 PI = Restricted (1), SI = Not Screened (0)
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62>; screen=no; party=calling; privacy=off  Proxy-Require: Privacy	Untrusted	Disable	No CPN	None	H.225 PI = Restricted (1), SI = Not Screened (0)
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62>; screen=no; party=calling; privacy=off  Proxy-Require: Privacy	Untrusted	Enable	No CPN	Nextone	H.225 PI = Restricted (1), SI = Not Screened (0)

**Table 42. SIP (Draft 01) to H.323, Caller ID blocking enabled**

INPUT			OUTPUT		
Incoming Privacy	Destination trust	Q.931 display	Calling Party Number (CPN)	Display IE	Privacy indicators
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62>; screen=no; party=calling; privacy=off  Proxy-Require: Privacy	Trusted	Disable	No CPN	None	H.225 PI = Restricted (1), SI = Not Screened (0)
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62>; screen=no; party=calling; privacy=off  Proxy-Require: Privacy	Trusted	Enable	No CPN	Anonymous	H.225 PI = Restricted (1), SI = Not Screened (0)
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62>; screen=no; party=calling; privacy=off  Proxy-Require: Privacy	Untrusted	Disable	No CPN	None	H.225 PI = Restricted (1), SI = Not Screened (0)
From: "Nextone" <sip:anonymous@localhost>  RPID:<sip:sipp@172.16.1.62>; screen=no; party=calling; privacy=off  Proxy-Require: Privacy	Untrusted	Enable	No CPN	Anonymous	H.225 PI = Restricted (1), SI = Not Screened (0)

## H.323 Services

*Note: Running H.323 services on iServer requires a feature license. See Chapter 5, MSX Licenses on page 100 for details on iServer licenses and procedures to obtain them.*

### H.323 call flows

H.323 call setup follows a strict sequence of steps that are tightly prescribed by the H.323 protocol definition. Good information on H.323 and its components is available through the IEC's discussion of it in their web site, at <http://www.iec.org/online/tutorials/h323/topic01.html>. That site also provides the complete protocol standard for reference. The information in this section is not intended as a complete discussion of H.323 call flows, but to give a basis for help in understanding the rest of the chapter.

#### **Registration and setup flow**

Figure 34 shows three possible call flows, to illustrate the ARQ option for H.323. The green numbers in the diagrams refer to the sequence in which the steps occur.

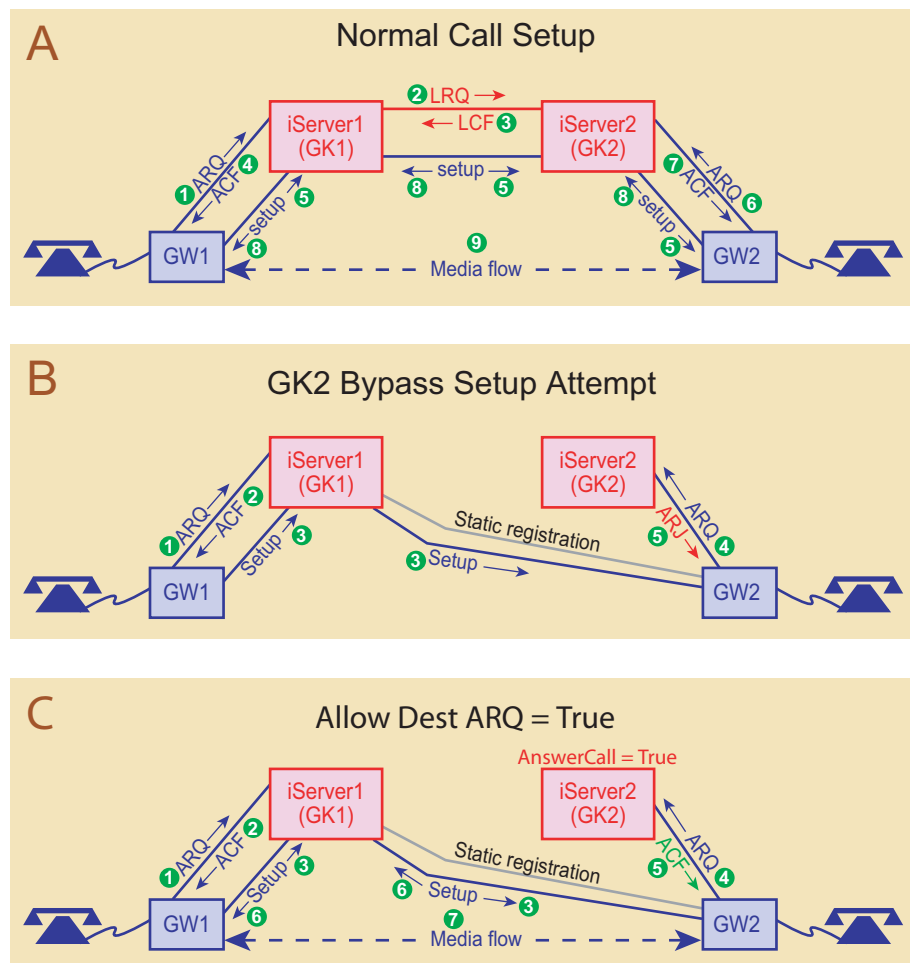
**Example A, Normal.** Ordinarily, for a call to be set up on two gateways registered to separate iServer-gatekeepers, the two iServers must be registered with each other. In such a case, the normal call setup proceeds from Gateway 1 to its Gatekeeper (iServer1), then to the second Gatekeeper (iServer2), which sets up the call with Gateway 2. Media then flows between the two gateways. The numbers in the diagram refer to flow sequence. (Note that the "setup" is not detailed here, so all setup activity in both directions is shown as one step.)

**Example B, GK2 Bypass.** If GW 1 tried to set up a call through iServer1 with GW 2 directly, without involving iServer2 (which is sometimes attempted for fraudulent reasons), when GW2 received the setup message, it would send an ARQ to the GK to which it is dynamically registered, iServer2. Because iServer2 doesn't know iServer1, iServer2 can't control the session or create a CDR, and it denies the access by sending an ARJ to GW2, which then refuses to set up the call.

**Example C, Allow Dest ARQ.** The Allow Dest ARQ option can be set on RSM Console's H.323 tab for iServer2. If this option is set, iServer2 responds to

GW2's ARQ with an ACF, and the call goes through, even though iServer2 cannot control the session. Normally, this situation is not desirable, but there can be cases where it's necessary to circumvent this protection.

**Figure 34. Example Call Flows**



The Allow Dest ARQ option can be set in two ways:

- **Preferred:** Set the option using the H.323 tab of the iServerConfiguration window in RSM Console.
- **Alternate:** Set the allow-destarq-all attribute to "enabled" using the nxconfig.pl utility.

To set the allow-destarq-all attribute to "enabled" using nxconfig.pl:

1. Log on to the iServer.
2. Execute the following commands:

```
nxconfig.pl -e allow-destarq-all -v 1
```

### **Clearing AROs through an Sgatekeeper**

iServer provides an option for clearing all ARQs received from unregistered endpoints by passing the request on to an Sgatekeeper for authorization.

To set the Allow Auth. ARQ option:

- **Preferred:** Set the option using the H.323 tab of the iServerConfiguration window in RSM Console.
- **Alternate:** Set the allow-autharq-all attribute to “enabled” using the nxconfig.pl utility.

To set the allow-autharq-all attribute to “enabled” using nxconfig.pl:

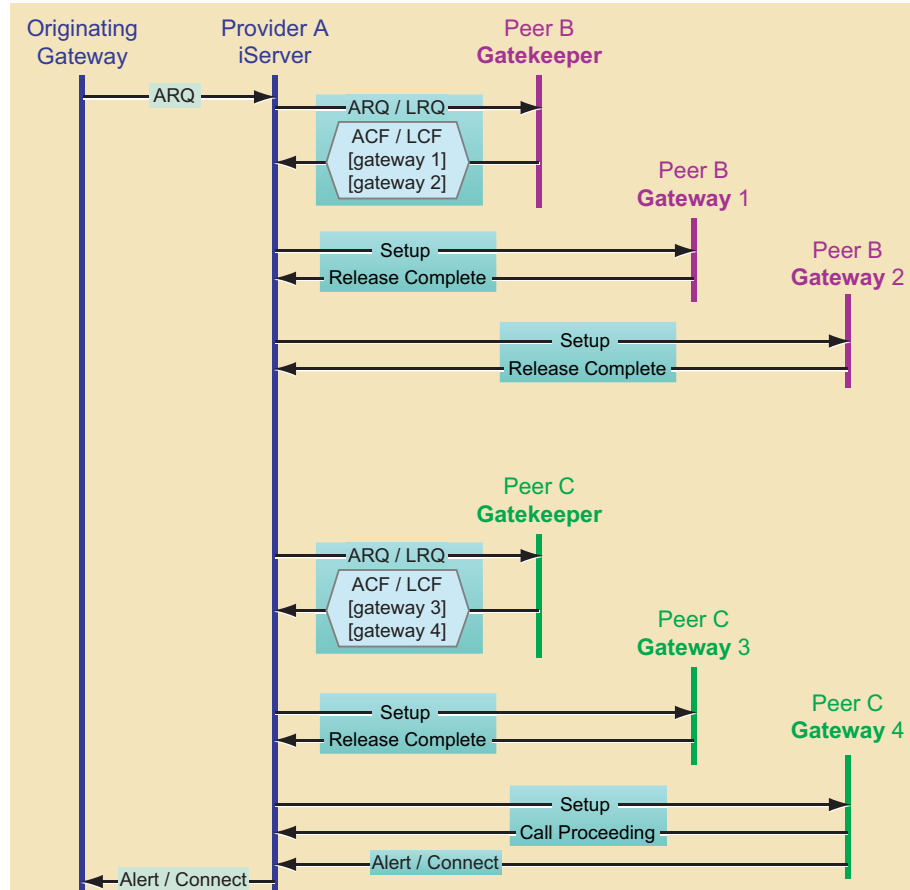
1. Log on to the iServer.
2. Execute the following commands:

```
nxconfig.pl -e allow-autharq-all -v 1
```

### Multiple routes to egress

Now, let's consider another possibility. Figure 35 depicts one simplified H.323 call setup scenario, where a request for admission results in multiple addresses returned from one or more gatekeepers.

**Figure 35. H.323 Call Setup with Multiple Addresses Returned**



In the scenario illustrated in this figure, three providers are involved, called Provider A, and Peers B and C. The sequence depicted is strictly followed, from top to bottom. The steps are:

1. Provider A's originating gateway sends an admission request (ARQ) to the iServer.
2. The iServer sends an ARQ or LRQ of its own to a gatekeeper belonging to one of its peering partners, Peer B.



3. Peer B's gatekeeper sends back one or more (in this example, two) addresses of gateways in its ACF/LCF, through which the call may be routed.
4. The iServer first tries to set up a call with Peer B's Gateway 1, and the setup does not complete (as shown by the *release complete* message).
5. The iServer then tries to set up the call through Peer B's Gateway 2, but that attempt fails, too.
6. Now the iServer sends a new ARQ or LRQ to Peer C's gatekeeper, which returns the addresses of two Peer C's own gateways.
7. The iServer sends a call setup request to Peer C's Gateway 3, and again the call fails through this gateway, with the gateway returning a *release complete*.
8. Finally, the iServer sends a call setup to Peer C's Gateway 4, and the call gets connected, with an alert (ring) and connect.

## iServer as an H.323 gatekeeper

In an H.323 network, the iServer is compliant with Version 3 suite (ITU-T H.323 suite) of standards, and can operate:

- As an H.323 gatekeeper
- As a domain controller

By default, the iServer is configured as an H.323 gatekeeper, and can be used for:

- Registration and Admission and Status (RAS) services
- Routing services
- Directory services
- Other services such as proxy as gateway, multiple domain support, interoperability with other gatekeepers and CDR support.
- Security

### **RAS services**

When an endpoint is configured on an iServer gatekeeper, it can be configured as a static or a dynamic endpoint. A dynamic endpoint is expected to register itself with the iServer gatekeeper.

- To add an endpoint to register, you can add one of its attributes (such as IP address, phone number, or H.323 ID), to the iServer database.
- To make an endpoint static, you must add the IP address of the static endpoint to the iServer database.

### **Alias types for registration**

You can dynamically register an endpoint with the iServer gatekeeper using the endpoint's H.323 ID. You can also add an endpoint as a virtual endpoint (i.e., even when the endpoint is just a "user" with a phone number and not a device) or as a static endpoint. A virtual endpoint can be used for presence/cut-through "Find me Follow me," provided it is configured with a phone number on the iServer.

***Note:** By default, to make a call to or from an endpoint in the network, it **MUST** be configured on the iServer. However, if you wish to allow calls from endpoints not configured on the iServer, you can enable the Allow All Sources option on RSM Console's System tab.*

### **Alternate gatekeeper support**

When an endpoint registers with an iServer (H.323) gatekeeper, the iServer can be configured to provide the endpoint with the IP address of an alternate gatekeeper. If the iServer becomes non-functional for some reason, the endpoint is automatically supported by the alternate gatekeeper for all registration and authentication services.

### **Registration request (RRQ)**

A dynamic endpoint in an H.323 network must send light weight RRQs to the iServer at regular intervals for the iServer to consider it *active*. When an endpoint first registers with an iServer (H.323) gatekeeper with a normal RRQ, the iServer indicates to the endpoint the interval that you have configured for light weight RRQs in the iServer configuration table. Subsequently, it expects an RRQ from the endpoint at the preset time interval. If it receives an RRQ before the interval lapses, it adds the endpoint to its *active* list. If not, it considers the endpoint inactive, although it still retains all provisioning and configuration information for the endpoint in its database. Statically-provisioned gateways are always considered active.

***Note:** To be able to receive a call, an endpoint must either be registered with the iServer, or must be configured as a static endpoint. Also, if it is not a static endpoint, it must be "active" for the iServer to route calls to it.*

If you specified a gatekeeper ID when configuring the iServer, all endpoints that use the iServer as a gatekeeper must supply that gatekeeper ID as part of

the RRQ they send to the iServer when they first request registration on it, in order for the iServer to respond.

If you leave the gatekeeper ID field blank, the iServer will not respond to any registration attempts from the gateway. Upon installation of the iServer, the field is blank, which is considered the default.

### **H.323: RAS buffer size**

While the default RAS instance buffer size of 2048 (2K) bytes is adequate for most applications, some large installations will require a larger buffer size to handle the oversize packets sent by some gateways, such as Lucent's TNT.

*Note: Changing this buffer size on systems with large call capacities can consume a great deal of system memory. For example, on a system with a 12,500 call maximum, each 1K increase to the RAS buffer size results in the additional allocation of 300MB of memory. Therefore, avoid increasing the size of this buffer, especially on systems with large call capacities, unless it is needed.*

To change the buffer size for RAS instances,

1. Log on to the iServer.
2. Set the H.323 RAS buffer size. Enter:

```
nxconfig.pl -e h323-maxrasbuffsize -v value
```

where value is the buffer size to set. Restart iServer when prompted.

*Note: In-process H.323 calls, all incomplete call setups, are dropped during a restart.*

Specify a value larger than the default, such as 3072 or 4096. 1 Kilobyte (1024-byte) boundaries are recommended.

### **Routing services**

By default, the iServer gatekeeper routes both H.225 and H.245 signaling for H.323 calls. You can configure the iServer to generate Call Detail Records (CDRs) for each call it routes in the network (see Chapter 21, *Billing and CDR Processing* on page 393). These CDRs can then be fed to a billing system to generate actual customer bills.

### **iServer as a domain controller**

The iServer (H.323) gatekeeper can be configured to act as a proxy for third party endpoints. In that mode, it registers on the endpoint's behalf with a third party gatekeeper. When operating in this mode, the iServer (H.323) gatekeeper appears as a call routing gatekeeper to endpoints registered with it, and as a gateway to the third party gatekeeper. As a proxy, it can also set up and control calls to other endpoints in the network. To use this feature, you must configure

the third party gatekeeper as an “Sgatekeeper” on the iServer using RSM Console. Refer to *About the sgatekeeper* on page 12 for a short description of the Sgatekeeper feature.

### **Calling plan support**

The iServer (H.323) gatekeeper supports calling plans (see Chapter 25, *Calling Plans* on page 466) and all the calling plan application scenarios detailed in the section *Other calling plan application scenarios* on page 487. It has the capability to translate dialed numbers to global numbers while routing calls using calling plans.

### **Tech prefix**

The iServer (H.323) gatekeeper supports Cisco tech prefixes. A tech prefix is an identifying tag used by a gatekeeper for gateway selection within the zone or domain it is controlling. The tech prefix is provided by the gateway when it registers with the gatekeeper. When a Cisco gatekeeper is configured as an Sgatekeeper (see *About the sgatekeeper* on page 12), the iServer can send a tech prefix along with the RRQ that it sends to register with the Sgatekeeper.

*Note: If a Cisco gateway presents a tech prefix when it registers with the iServer, the iServer ignores the tech prefix (along with its E.164 number). Instead, it matches the Cisco gateway’s aliases (such as H.323 ID) with the credentials configured for the gateway in the iServer database.*

For compatibility with Cisco gatekeepers and gateways, the iServer can also use the tech prefix that is configured as part of a calling plan, to determine the Cisco gateway that it must route a call to. For a description of how the iServer uses a tech prefix in a calling plan, refer to *Source tagging* on page 479.

### **Cisco gatekeeper clustering**

The iServer system provides features to facilitate interoperability with Cisco’s Gatekeeper Cluster scheme. This scheme offers some advantages, including better location lookup (LRQ) performance. As a result of these changes, the iServer can now reside on the edge of an H.323 network that also includes Cisco gatekeepers.

For any gatekeeper to cluster with a Cisco GK, the peering gatekeeper must send a “time-to-live”, or TTL value with its registration request (RRQ). TTL is a method by which gatekeepers keep track of which peering gatekeepers are currently operational. Here’s how it works:

1. A gatekeeper (we'll call it GK2) sends a Registration Request (RRQ) to another gatekeeper (GK1), that is a Cisco gatekeeper. The default TTL value for a Cisco gatekeeper is 1800 seconds (30 minutes).
2. If GK2's RRQ contains no TTL value, LRQs sent to GK1 will just be dropped, without even an LRJ from GK1.
3. GK1 expects to hear from GK2 periodically, but never longer than the TTL time GK2 specified in its RRQ. If GK1 doesn't hear from GK2 within that time frame, it begins querying GK2 with Information Request (IRQ) messages. After a time configured in GK1, if GK1 still hasn't heard from GK2, it will remove GK2's entry from its registered gatekeeper list.

The TTL value for an iServer is factory-set at 6 minutes, and cannot be changed.

### **Directory services**

The iServer (H.323) gatekeeper maintains a directory for all static and registered endpoints on the H.323 network. You can also store other information about endpoints (such as user name and location) in this directory. The iServer uses this information for authenticating every call that is initiated in the network.

You can also add endpoints using an H.323 ID on the iServer using RSM Console. See RSM Console online help for the detailed procedure to do this.

The iServer can be configured to be a directory gatekeeper for the domain controller. You can configure the iServer in this mode by configuring the domain controller on the iServer as an endpoint of type xGatekeeper. When operating in this mode, the iServer sends and responds to LRQs.

In some cases, a gatekeeper's response to an LRQ or ARQ may contain more than one address. iServer correctly handles this situation by hunting through the list of returned gateways until the call is completed, or the entire list has been exhausted. See *Multiple routes to egress* on page 264 for details.

### **Other H.323 services**

The iServer (H.323) gatekeeper can also:

- **Proxy as a gateway**—when inter-operating with other gatekeepers, the iServer can emulate a gateway registered with the gatekeepers and routes calls to and from them. Alternately, it can do gatekeeper to gatekeeper peering using LRQs.

- **Provide multiple domain support**—you can configure multiple domains with different VPN IDs, or multiple groups with similar VPN IDs on the same iServer gatekeeper machine.
- **Interoperate with other gatekeepers**—including Clarent, Cisco, RadVision, and Vocaltec.
- **Operate in stateless mode**—when configured for stateless mode, the iServer acts as a directory gatekeeper responding only to RAS messages. It does not generate CDRs in when configured in this mode. To configure the iServer in this mode, you must have both H.323 call routing and H.245 routing disabled. For the procedure to set these parameters, refer to *Using nxconfig.pl to configure H.323 optional parameters* on page 275.

## H.323 protocol support

**Table 43. H.323 Protocol Support**

Feature	Supported?	Notes
Tech prefix	Yes	If a Cisco gateway presents a tech prefix when it registers with the iServer, the iServer ignores the tech prefix (along with its E.164 number). Instead, it matches the Cisco gateway's aliases (such as H.323 ID) with the credentials configured for the gateway in the iServer database.
Tech prefix gateway	Yes	The iServer can eliminate the tech prefix given by a gateway, in the setup message that it sends to the destination gateway.
LRQ	Yes	For address resolution, the iServer can send LRQs to endpoints configured as xgatekeepers. It also responds to an LRQ with an LCF/LRJ.

Feature	Supported?	Notes
Codecs	Yes	<p>The iServer supports:</p> <ul style="list-style-type: none"> <li>• G.711a-law</li> <li>• G.711<math>\mu</math>-law</li> <li>• G.723</li> <li>• G.728</li> <li>• G.729</li> <li>• G.729a</li> <li>• G.729b</li> <li>• G.729awb</li> </ul> <p>Limited support (for H.323-to-H.323 calls only) is also provided for:</p> <ul style="list-style-type: none"> <li>• 7.23AR 53 &amp; 63</li> </ul>
Null TCS (terminal capability)	Yes	The iServer closes logical channels on receiving Null TCS.
Gatekeeper routed signaling	Yes	By default, the iServer does gatekeeper routed call signaling.
Direct routed signaling	Yes	The iServer can be configured for operation in this mode.
Fast start	Yes (this option can be disabled)	If the iServer receives a fast start setup it forwards it in the outgoing setup. It also forwards fast start responses that it receives in Call Proceeding/Alerting/Connect messages.
H.245 routing	Yes	The iServer can be configured to route H.245 calls. When configured for this feature, the iServer is involved in all H.245 message exchanges.
H.245 tunneling	Yes	The iServer can be configured to support H.245 tunneling.
H.450	No	

Feature	Supported?	Notes
GRQ	Yes	When acting as a gateway registered with a third party gatekeeper, the iServer can be configured to send a GRQ for gatekeeper discovery (see page 73). The iServer (H.323) gatekeeper also responds to an incoming GRQ with a GCF.  <i>Note: The iServer only supports GRQs unicast to the iServer's realm on port 1719. Broadcast and multicast GRQs are not supported.</i>
H.235	Yes	Configuring a password for an endpoint enables H.235 for that endpoint. The iServer supports the authentication mechanism of a password with hash. It uses the MD5 algorithm.
ARQ	Yes	The iServer responds to an ARQ with an ACF/ARJ. It can send an ARQ to a third party gatekeeper when acting as a gateway.
RRQ/light weight RRQ	Yes	The iServer sends an RCF for every RRQ it receives. It can also send an RRQ when registering with a third party gatekeeper.
BRQ/BCF	Yes	
DRQ/DCF	Yes	
RAI	Yes	The iServer (H.323) gatekeeper sends an RAC in response to every RAI it receives. If an endpoint indicates that it is almost out of resources, the iServer stops routing calls to it. When configured as an H.323 gateway, the iServer can send an RAI to a third party gatekeeper.
T-38 fax	Yes	
Annex D	Yes	
ISDN call progress	Yes	
ISDN cause codes	Yes	The iServer relays ISDN cause codes from one call leg to the other.



## Configuring the iServer as an H.323 gateway

By default, the iServer acts as an H.323 gatekeeper, so that endpoints can register with it. However, if required, the iServer can also act as an H.323 gateway, registering with a third-party gatekeeper that is configured as a Master Gatekeeper (MGK), known in some contexts as an *Sgatekeeper* (“super gatekeeper”).

To configure the iServer as an H.323 gateway, you must do the following using RSM Console:

- Configure the third-party gatekeeper as an endpoint of the type *Sgatekeeper*.
- Register the iServer (H.323 gateway) endpoint with the *Sgatekeeper* using its H.323 ID.

*Note: Before you register the iServer (H.323 gateway) with the third party Sgatekeeper, you must provision it on the Sgatekeeper using the procedure(s) recommended by the third party Sgatekeeper vendor.*

Once registered with the *Sgatekeeper*, the iServer (H.323 gateway) sends an RRQ (and/or lightweight RRQs) and other RAS messages to the *Sgatekeeper*.

### **Alternate gatekeeper support for an Sgatekeeper**

When using the iServer as a gateway, you can configure a static alternate gatekeeper for each *Sgatekeeper* that you configure. If the *Sgatekeeper* ceases to operate for some reason, the iServer will use the alternate gatekeeper for sending ARQs.

### **Configuring an alternate gatekeeper for an Sgatekeeper**

To configure an alternate gatekeeper for an *Sgatekeeper*, you must do the following using RSM Console:

1. Add an additional port for the *Sgatekeeper* that you already configured, using the same registration ID.
2. Enter the following information for this port:
  - IP address
  - Phone number
  - H.323 ID
  - Tech prefix (see *Tech prefix* on page 268 for more information on this variable)

*Note: For the iServer to use a port as an alternate Sgatekeeper, the port must be statically added.*

You may add as many ports for the Sgatekeeper under the same registration ID. However, the iServer registers with only one of the ports at a time.

The iServer attempts to register with a port, going serially from port 1 to port 2 to port 3, and so on. As soon as it encounters a port that it is able to register with, it registers with that port. Once registered with a port, the iServer uses the IP address configured with that port as the IP address for the entire cluster of configured ports. It stays registered with this port until:

1. It receives a URQ from the port
2. It receives an RRJ from the port
3. The RRQ times out

When this port ceases to operate, the iServer moves to the next port in serial order, and attempts to register with it. Thus, every time an Sgatekeeper port ceases to operate, the iServer keeps the Sgatekeeper active by moving to the next available port.

*Note: In order to facilitate calls in using the Sgatekeeper, the iServer **MUST** register with it **BEFORE** a call is initiated.*

## Information transfer capability

The iServer can restrict the kinds of digital information that network endpoints will carry. The *servercfg* table has an attribute, *h323-infotranscap*, that provides the default transfer capability value for endpoints. This can be one of several capabilities, listed in Table 44 and assigned using the *nxconfig.pl* utility. In addition, with RSM Console the user can configure any endpoint to override the global setting.

**Table 44. Information Transfer Capability Options**

Option Name	Description
speech	Normal speech traffic
unrestricted	Unrestricted digital information
restricted	Digital information in which restrictions apply to the bit formats
audio	3.1 KHz audio
unrestrictedtones	Unrestricted digital information, with tones/announcements
video	Video data is permitted

Option Name	Description
pass	(Default) Pass through whatever is received from the terminal

To edit the `h323-infotranscap` value:

1. Log on to the iServer.
2. Enter:

```
nxconfig.pl -e h323-infotranscap -v value
```

where value is one of the values listed in the “Option Names” column of Table 44.

Note that for an endpoint to use this global default value, that endpoint must have “default” set as its “Info Transfer Cap” value.

## Configuring H.323 optional parameters

You can use optional H.323 services (such as H.323 fast start, force H.245, and H.245 routing) in your network. To use these services, you must configure both the iServer and the endpoints that need to use these services, for H.323.

### Enabling H.323 on the endpoints

Any endpoint in the network can be configured for H.323, provided the iServer it is registered with has a valid license file with H.323 enabled. Refer to RSM Console online help for the detailed procedure to enable H.323 services on an endpoint.

### Using `nxconfig.pl` to configure H.323 optional parameters

All global H.323 parameters for the iServer are stored in the `servercfg` table on the iServer. The `nxconfig.pl` utility is used to modify this table. See Chapter 3, *MSX Configuration* on page 10 for a description of the `servercfg` table and the `nxconfig.pl` utility.

The `nxconfig.pl` configures the H.323 global parameters listed below. Some of these parameters can be overridden for individual endpoints.

- The number of H.323 instances to run on the server
- The maximum number of allowable H.323 call setups (ARQs) per second
- The maximum number of simultaneous H.323 calls<sup>1</sup>

- Default global H.323 Information Transfer Capability (speech, unrestricted, restricted, audio, pass, etc.), as described in *Information transfer capability* on page 274.
- H.323 call routing
- H.323 fast start
- H.245 routing
- Hairpin calls
- Local proceeding
- H.245 tunneling
- RFC 2833 capability (DTMF via RTP)
- T38 Annex D fax capability

Edit the settings for these parameters using the `nxconfig.pl` utility. Before running the `nxconfig.pl` utility, follow these steps, then see the procedure corresponding to the H.323 parameter you want to set, below.

1. Log on to the iServer.
2. Enter the command shown in the applicable subsection below.

#### **Set the number of H.323 instances to run**

You can change the number of concurrent instances of the H.323 stack to run from its default value of 1. NexTone recommends leaving it at “1” (though you may set it to another value during troubleshooting, if directed to do so by NexTone Support). To set this value, enter:

```
nxconfig.pl -e h323-instance -v value
```

where value is the number of instances you want to set. Restart the iServer when prompted.

**Note:** *In-process H.323 calls, all incomplete call setups, are dropped during a restart.*

#### **Set the number of H.323 calls per second**

You can set the H.323 calls-per-second (ARQs). A value other than the default may be used to control the throttling of incoming ARQs from call origination endpoints. Any ARQs which are in excess of this per-second rate are rejected with a “resource unavailable” response. Rejections are recorded in CDRs. To set this value, enter:

1. This parameter controls only the *total* number of H.323 calls. Individual endpoints can override this setting, and set limits based on ingress and egress. For details, see *Setting session limits on calls, by endpoint* on page 79.

```
nxconfig.pl -e h323cps -v value
```

where value is the number of calls per second to set.

***Caution: This setting should only be changed as directed by NexTone Support.***

### **Set the number of H.323 maximum calls**

The H.323 Maximum Calls parameter configures the H.323 stack for the combined total number of incoming and outgoing H.323 call legs. All resources specified here are pre-allocated by the system at startup time. Remember that each endpoint can override this combined limit, and place other limits on ingress and egress calls.

Note that the recommended initial value for this parameter is 2.5 times the number of licensed vports for this iServer. To set this value, enter:

```
nxconfig.pl -e h323-maxcalls -v value
```

where value is the maximum calls value to set. Restart the iServer when prompted.

***Note: In-process H.323 calls, all incomplete call setups, are dropped during a restart.***

### **Enabling H.323 call routing**

To enable or disable H.323 routed calls, enter:

```
nxconfig.pl -e route-call -v value
```

where value is 1 to enable or 0 to disable the feature.

### **Enabling H.323 fast start**

Setting this attribute to “enabled” sends fast start setups to terminating H.323 endpoints. Enable or disable this feature by entering:

```
nxconfig.pl -e faststart -v value
```

where value is 1 to enable or 0 to disable the feature.

### **Enabling H.245 Routing**

This attribute setting enables/disables H.245 routing on the iServer. Enable or disable this feature by entering:

```
nxconfig.pl -e route-h245 -v value
```

where value is 1 to enable or 0 to disable the feature.

### **Enabling Hairpin Calls**

Hairpin calls are calls that have the source and destination in the same gateway. Enable or disable this feature by entering:

```
nxconfig.pl -e hairpin -v value
```

where value is 1 to enable or 0 to disable the feature.

### **Enable/Disable Local Proceeding**

iServer installations where call hunting is used may want Local Proceeding enabled. For more information, see *Configurable local proceeding* on page 280. Enable or disable this feature by entering:

```
nxconfig.pl -e local-proceeding -v value
```

where value is 1 to enable or 0 to disable the feature.

### **Enable/Disable H.245 Tunneling**

The iServer system is configured to support H.245 tunneling. Enable or disable this feature by entering:

```
nxconfig.pl -e h245-tunneling -v value
```

where value is 1 to enable or 0 to disable the feature.

### **Remove Assertion of RFC 2833 Codec Capability**

To remove assertion of rfc2833 codec capability during a “fast start,” thereby allowing for the negotiation of it, set this attribute to *enabled*. Otherwise, set it to *disabled*. Enable or disable this feature by entering:

```
nxconfig.pl -e h323-removetcs2833 -v value
```

where value is 1 to enable or 0 to disable the feature.

### **Remove Assertion of TCS T38 Fax Capability**

To remove assertion of TCS T38 fax capability during a “fast start,” thereby allowing for the negotiation of it, set this attribute to *enabled*. Otherwise, set it to *disabled*. Enable or disable this feature by entering:

```
nxconfig.pl -e h323-removetcs38 -v value
```

where value is 1 to enable or 0 to disable the feature.

## **H.323 trunk group support**

iServer supports trunk-based routing. Details on that capability are given in the *Calling Plans* chapter, under the heading, *iServer trunk group support* on page 503.

Note that an H.323 setup request from an endpoint can include fields representing its source and destination trunk groups. These fields are known in the H.323 setup as `sourceCircuitID` and `destinationCircuitID`. They map to the

configured endpoint's Src. Trunk Group and Dest. Trunk Group fields, respectively, in the iServer.

## H.235 security authentication

The iServer supports H.235 security authentication in both modes: while acting as a gatekeeper and while acting as a gateway. The mechanism that it supports is pwdHash (password with hash) and the algorithm is MD5. This complies with the iNow gateway authentication recommendation and is interoperable with most commonly deployed vendors including Cisco and VocalTec.

### **Registering to a gatekeeper using H.235**

When the iServer is configured to act as a gateway, it is capable of registering to a gatekeeper using the H.235 security mechanism.

The iServer sends out GRQ with authenticationCapability set to pwdHash. The algorithmOIDs must be set to the object ID for the MD5 algorithm { 1 2 840 113549 2 5 }.

In the RRQ message, the iServer adds MD5 hash of ASN.1 encoded cryptoToken for authentication. The cryptoToken contains the following:

- H.323 ID of the iServer
- timestamp
- password.

### **Configuring H.235 security authentication**

Complete the following steps to configure H.235 security authentication.

1. Configure the H323ID which the iServer will use to register with the gatekeeper.
2. Configure the password that it will use for authentication with the gatekeeper.
3. Use the NTP to synchronize the time between the iServer and the gatekeeper.
4. If registering to multiple gatekeepers using H.235 authentication, ensure that all the gatekeepers have the same time.

**Note:** *H.235 security authentication is enabled only if the H.235 security password is configured in the configuration of the Sgatekeeper.*

### **Logging support**

H.323 logging is done using the Unix syslog facility. To enable H.323 logging, enable level 3 logging in the `syslog.conf` file, then restart `syslogd` for it to take effect. To restart `syslogd`:

1. Log on to the iServer as root.
2. Restart `syslogd`. Enter:

```
/etc/init.d/syslog restart
```

### **Configurable local proceeding**

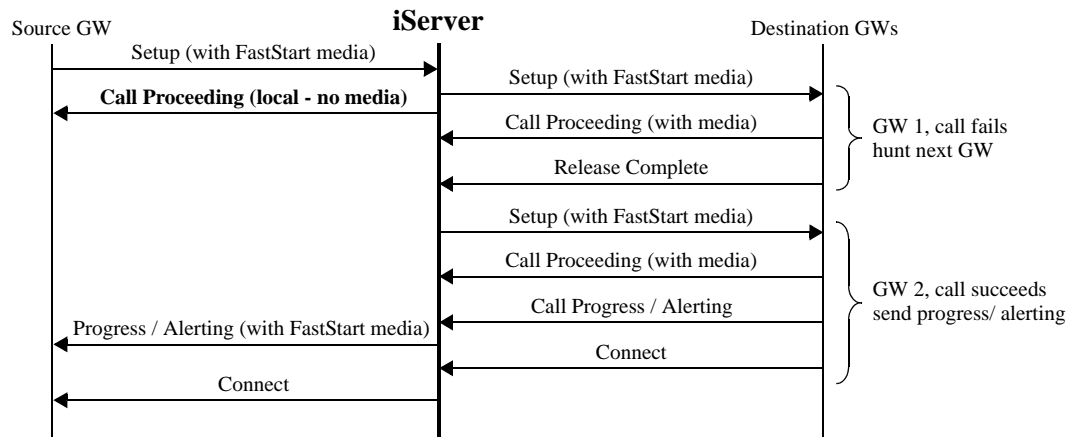
This is a global configuration available to enable sending a local call proceeding message. By default, the behavior of the iServer is to relay call proceeding messages from the egress leg to the ingress leg. However, certain source endpoints may time out waiting for the call proceeding indicator, especially if the iServer is performing call hunting. In these situations, this global parameter for generating local proceeding can be enabled. The iServer then generates Call Proceeding messages back to the source endpoint. (This parameter only applies to H.323 setups).

In typical situations (without Local Proceeding), the destination GW sends a Call Proceeding to the iServer, which passes it back to the source. If the Call Proceeding contains FastStart (Codecs and RTP port info), the iServer relays the Call Proceeding to the source device as well. This scenario prevents hunting functionality.



Figure 36 illustrates the Local Proceeding state diagram.

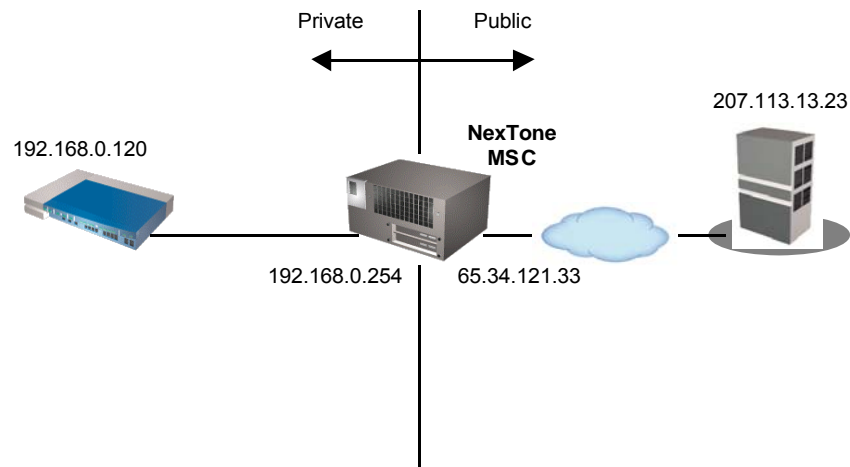
**Figure 36. Local Proceeding Flow Diagram**



### **Registrations from public and private networks**

If FCE (see *Firewall Control Entity (FCE)* on page 334) is enabled, the iServer supports incoming registrations from both public (routable) and private (non-routable) networks. Figure 37 illustrates this concept:

**Figure 37. Registrations from Private and Public Networks**



In the configuration shown in Figure 37, the media flows through the iServer when the call gets setup from the private to public network or vice versa. The logical areas from which an endpoint may register are known as *realms*. For details on realms and how they are implemented on the iServer, see the chapter, *MSX Realms* on page 174.

### **Optional H.245 address in connect message**

When setting up calls from the iServer to certain gateways, there is a chance of getting into a *glare* condition when setting up the H.245 connection. This is because the H.245 address is sent from the iServer in the CONNECT message and in a separate FACILITY message. To avoid such glare conditions, the iServer can be configured to send H.245 address only in a separate FACILITY message and not in the CONNECT message.

This configuration is available for endpoints, via RSM Console. To set this parameter:

1. From the RSM Console main screen, right-click the server that the endpoint resides on, and choose **Properties**.
2. From the **Database** window, locate the endpoint or create a new one.
3. On the **Modify** (or **Provision**) endpoint window, click the **Protocol** tab.
4. Click the **H.323 Configure** button. The **H.323 Protocol Parameters** window appears.
5. Select the **H.245 Address in Connect** option to allow the address to be sent in CONNECT messages. Deselect the option to prevent sending the address in CONNECT messages, thereby preventing the above-described glare condition.

### **Call progress indication**

Call progress indication for calls flowing between H.323 endpoints is now supported. A call progress message from one leg of the call is relayed to the other leg. No endpoint-specific configuration is necessary to enable this feature.

### **Configurable Bearer Capability (layer 1 protocol)**

Bearer capability from the ingress leg can be configured to be relayed to the egress leg during call setup. In earlier versions of the iServer, the bearer capability on the egress leg was always G.711ulaw. This can be configured now. The configuration is specific to destination endpoints, so that the iServer

can use the appropriate bearer capability during the egress leg call attempt. Default is G711 A-law.

The bearer capability values possible are:

- Passthrough
- G711 u-law
- G711 A-law
- H.221

This configuration is available in RSM Console and is set on destination endpoints. The “Information Transfer Capability” field is always set to “Speech” for all egress call setups irrespective of the value of this parameter received in the ingress call setup.

### ***How to configure Bearer Capability (layer 1 protocol)***

This configuration is available for H.323 endpoints, via RSM Console. To set this parameter:

1. From the RSM Console main screen, right-click the server that the endpoint resides on, and choose Properties.
2. From the **Database** window, locate the endpoint or create a new one.
3. On the **Modify** (or **Provision**) endpoint window, click the Protocol tab.
4. Click the H.323 Configure button. The **H.323 Protocol Parameters** window appears.
5. Click the Layer 1 Protocol pull-down menu, and select the appropriate option. Pass-Through means that the capability from the ingress leg is relayed to the egress leg.

### ***Configurable Party Number Type***

You can provision an egress gateway so that the Called Party Number Type, or Calling Party Number Type, on the egress leg of a call setup sent to it is forced to International, National, Unknown or another supported value. Setting this parameter to Pass (the default setting) forwards the party number type received on the call’s ingress leg. Otherwise, calls sent to the gateway are marked with the number type you select.

Supported number types are:

- Pass (default)
- Unknown

- International
- National
- Specific
- Subscriber
- Abbreviated

### **How to configure Number Type**

This configuration is available for H.323 endpoints, via RSM Console. To set this parameter:

1. From the RSM Console main screen, right-click the server that the endpoint resides on, and choose **Properties**.
2. From the **Database** window, locate the endpoint or create a new one.
3. On the **Modify** (or **Provision**) endpoint window, click the **Protocol** tab.
4. Click the H.323 Configure button. The **H.323 Protocol Parameters** window appears.
5. Click the Calling Party Number Type pull-down menu, and select the desired option.
6. Click the Called Party Number Type pull-down menu, and select the desired option.
7. Click OK to save your changes and close the **H.323** window. Click OK again to close the **Modify Endpoint** window.

### ***H.245 Capabilities Mode Restriction***

RFC 2833 defines how tones are handled by the RTP protocol. During capability negotiation, the H.245 TCS (terminal capability set) may indicate RFC 2833, and/or T38 fax carrying ability. Some receiving devices will not set up calls if these capabilities are indicated in the TCS.

The iServer has two global attributes related to suppressing indication of these services at the uport level. They are `h323-removetct38` and `h323-removetcs2833`, and they can be assigned settings of “enabled” or “disabled” through `nxconfig.pl`. By default, every endpoint inherits the setting from these global parameters, unless the settings for an endpoint are overridden as details in *How to configure H.245 Capabilities Mode Restriction*.

### **How to configure H.245 Capabilities Mode Restriction**

This configuration is available for H.323 endpoints, via RSM Console. To set this parameter:

1. From the RSM Console main screen, right-click the server that the endpoint resides on, and choose **Properties**.
2. From the **Database** window, locate the endpoint or create a new one.
3. On the **Modify** (or **Provision**) endpoint window, click the **Protocol** tab.
4. Click the **H.323 Configure** button. The **H.323 Protocol Parameters** window appears.
5. Scroll down to reveal the **Remove from TCS** frame if necessary. Using the **RFC 2833** and **T.38** pull-down menus, select the appropriate **enable** or **disable** options for this endpoint:
  - **default** – inherit the default, global setting
  - **enable** – do not send the capability, even if it does exist, or
  - **disable** – indicate the capability, if it exists

### **Vocaltec support considerations**

A few issues require special attention when implementing Vocaltec Gateways.

#### **H.235 Token Passing**

Vocaltec gatekeepers may also support H.235 token passing in the network.

***Note:** If this configuration is enabled, then it is imperative that the iServer host machine is a Network Time Protocol (NTP) client to the same NTP server as the Vocaltec gatekeeper. Refer to the manual page on `ntpdate(1)`.*

#### **Vocaltec gatekeeper configuration**

Given below are the steps involved in configuring a Vocaltec GK using RSM Console.

1. Synchronize the iServer platform with the same NTP Server as the Vocaltec gateway, using the command, `ntpdate NTP server IP`.

- Using RSM Console, add the Vocaltec gatekeeper as type “Master Gatekeeper.” Apply a Calling Plan, if available at this point.

**Figure 38. Adding a VocalTec Gatekeeper (Phone tab)**

The screenshot shows the 'Provision an Endpoint' dialog box with the 'Phone' tab selected. The configuration fields are as follows:

- Partition:
- Device Type:
- Registration ID:
- Port Number:
- IP Address:
- Extension:
- Phone Number:
- VPN Phone Number:
- Calling Plan:
- Realm:
- IEdge Group:
- ☐ Enable transcoding
- Codec Profile:

At the bottom are 'OK' and 'Cancel' buttons.

- Under the “Advanced” tab specify the subnet to which other source GWs (including VocalTec gateways) belong. Without this configuration, the iServer will reject H.225 signal requests from any source GW under the control of this Vocaltec gateway. If these GWs are spread across multiple subnets, then add extra ports to this GK configuration and specify as many

subnets as needed to cover all the GWs. For simplicity, NexTone recommends limiting this port addition to the minimum possible.

**Figure 39. Adding a VocalTec Gatekeeper (Advanced tab)**

The screenshot shows the 'Provision an Endpoint' dialog box with the 'Advanced' tab selected. The 'Phone Configuration' section contains the following fields:

- VPN Name:** (empty text field)
- VPN Group:** (empty text field)
- Zone:** 1 (text field)
- Vendor:** VocalTec (dropdown menu)
- Subnet IP:** 192.168.10.0 (text field)
- Subnet Mask:** 255.255.255.0 (text field)
- Outgoing Prefix:** (empty text field)

The 'Subnet' section, including the Subnet IP and Subnet Mask fields, is highlighted with a grey oval. At the bottom of the dialog are 'OK' and 'Cancel' buttons.

- Complete the H.323 protocol settings. Most VocalTec implementations use the format gatewayH323ID@VT\_GK\_DOMAIN.com to assign the H.323 ID to any gateway attempting to register. GRQ prior to RRQ is optional, but it should be enabled if RRJ with reason “DiscoveryRequired” are being received. H.235 security is turned on automatically by typing a

password string into the H.235 Password field. This and other gatekeeper-specific fields are highlighted in Figure 40.

**Figure 40. Adding a VocalTec Gatekeeper (Protocol -> H.323 Configure)**

**H.323 Protocol Parameters**

H.323

☒ GRQ

☐ RAI

☐ Apply Egress Routes to RAS Only

Tech Prefix:

H.323 Id:

Gatekeeper ID:

RAS Port:

Q.931 Port:

Calling Party Number Type:

Called Party Number Type:

Layer1 Protocol:

Info Transfer Cap:

H.235 Password:

☐ Q.931 Display

☐ Force H.245

OK Cancel

5. Additional parameters such as Max call threshold, Media Routing (if supported in license and platform) can be configured using the “Calls” tab.



## Inter-Working Function (IWF) Services

*Note: Running IWF services on the MSX requires a feature license. If H.323 and SIP services are enabled in the license file, this feature is automatically enabled.*

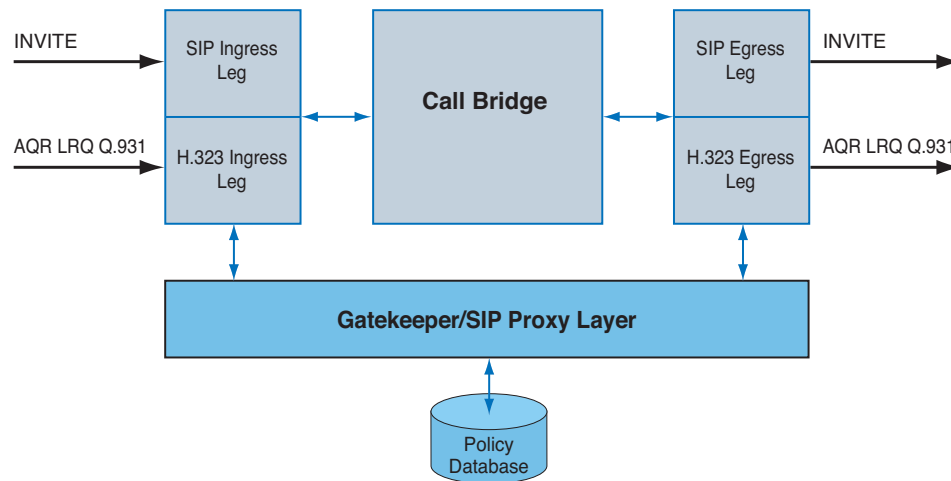
### What is IWF?

H.323 and SIP are the primary protocols for IP telephony in use today. Both protocols set up calls or sessions between IP telephony endpoints. H.323 and SIP specify media to be transported as Real Time Protocol (RTP). Thus, once the call signaling sets up the session, media between the endpoints flows as RTP directly between them. Bridging between H.323 and SIP protocols essentially becomes a problem for the signaling domain. The Inter Working Function (IWF) is an implementation of this bridging functionality between H.323 and SIP on the iServer.

MSX implements the IWF using the IETF draft *draft-agrawal-roy-palawat-sip-h323-interworking-03*.

### Signaling interworking and interoperability

The MSX provides both protocol and terminal interworking. The MSX implements interworking functions to bridge between the H.323 and SIP call setup protocols. This lets the MSX be neutral as to the call setup protocol which an endpoint uses to initiate a session or call. The benefit of this is that the MSX becomes a border control element for a multi-network deployment. From an architectural perspective, the MSX is both an H.323 gatekeeper and a SIP proxy server. As such, the MSX remains in the call signaling path while providing interworking but it does not participate in the media flow between endpoints. As part of terminal interworking, the MSX mediates between the differing capabilities exhibited by endpoint terminals.

**Figure 41. NexTone MSX Architecture**

This feature is useful for network service providers. By resolving the signaling conflict at the edge of the network, a network provider can successfully interconnect to any VoIP network while building towards a next generation SIP-based architecture. The MSX provides mediation for a number of critical functions. The enhanced H.323 gatekeeper functionality of the MSX allows carriers to seamlessly operate their existing mixed-vendor H.323 networks as a single network. In addition, the switch also provides a stateful SIP proxy server and a SIP/H.323 interworking function. This capability gives carriers a simple way to bridge their existing H.323 infrastructure and next generation softswitches and media gateways. In addition, the MSX acts as a policy enforcement point and controller for voice firewalls deployed at the edge of the network.

## Supported features and protocols

Table 45 and Table 46 list the IWF features and protocols that the MSX supports.

**Table 45. IWF Feature Support**

Feature	Support
T.38 fax	Yes
HOLD, and Hold with music	Yes
Transfer using Re-INVITE	Yes
Transfer using REFER	No

**Table 46. IWF Protocol Support**

Feature	Support
INVITE with SDP	Yes
INVITE without SDP	No
FastConnect with H.245	Yes
FastConnect without H.245	Yes
Non-FastConnect	Yes

## Cisco Call Manager – Sonus GSX interoperability

IWF calls between Cisco Call Manager (CCM) (H.323 Gateway, Non-FastStart) and Sonus GSX (SIP Gateway) are possible. To enable this feature, the Sonus GSX must be provisioned in the MSX database as a SIP gateway. A Sonus GSX can also be an H.323 gateway. The CCM version used in IWF is currently 3.2 (2c). The CCM is added as a static H.323 gateway that does not register with the MSX.

## Configuring the MSX for IWF

For IWF to become operational on your network, you must do the following:

1. Ensure that your license has both H.323 and SIP support enabled.

2. Use RSM Console to enable the codecs you want supported in your network(s). Refer to RSM Console online help for a detailed procedure on how to enable codecs.
3. Configure the MSX for SIP in stateful mode using `sconfig`.

## Forwarding signaling address for IWF Calls

A new configuration option forwards the H.323 signaling address of the source into the “From:” header of the SIP INVITE associated with the egress leg. This global parameter affects all IWF calls. This parameter is configured using `nxconfig.pl`.

## Disabling g729 variants for IWF calls

To ensure that calls are set up successfully between the SIP and H.323 endpoints, they must negotiate to use a codec that is supported by both. Because there are variants of the g729 codec, you might need to disable sending g729 variants on a SIP INVITE. This prevents a situation where the receiving endpoint negotiates to use a g729 variant and the originating endpoint supports only g729 thereby causing a codec mismatch. To ensure that both endpoints negotiate using the base g729 codec, a new option is available on SIP endpoints. When enabled, this option instructs the MSX to forward only g729 when it receives a SIP INVITE from that endpoint rather than variants such as g729 annex A or B. Use the following command:

```
cli iedge edit <regid> <uport> nog729variants [enable|disable]
```

## IWF and DTMF Translations

The MSX processes incoming messages with DTMF tones differently according to whether the messages are received as out-of-band (signaling) or in-band (media). When the input is out-of-band, it also bases its processing on whether the messages are SIP or H.323. SIP messages can be either INFO or NOTIFY. H.323 messages can be H.245 UII, H.245 alpha, or Q.931 INFO.

The type of input message and the configuration of the endpoint receiving the messages determines what the outgoing leg contains. The following table summarizes the types of conversions the MSX supports. “Yes” means the MSX performs the conversion between the ingress leg and the egress leg. “Relay” means the MSX passes the message unchanged. “IWF Relay” means the MSX passes the message through its Interworking Functions services. “No” means the conversion is not currently supported.

**Table 47. DTMF Translation Matrix**

<b>Egress Leg</b>	<b>INFO</b>	<b>NOTIFY</b>	<b>H.323 UII(S)</b>	<b>H.323 UII(A)</b>	<b>Q.931</b>	<b>RFC2833</b>	<b>G.711</b>
<b>Ingress Leg</b>							
<b>INFO</b>	Relay	YES	IWF Relay	IWF Relay	IWF Relay (Avaya)	YES	YES
<b>NOTIFY</b>	YES	Relay	YES	NO	YES	YES	YES
<b>H.323 UII(S)</b>	IWF Relay	YES	Relay	Relay	YES (Avaya)	YES	YES
<b>H.323 (UIIA)</b>	IWF Relay	YES	YES	Relay	YES (Avaya)	YES	YES
<b>Q.931</b>	IWF Relay (Avaya)	YES	YES	YES	Relay (Avaya)	YES	YES
<b>RFC2833</b>	YES	YES	YES	YES	YES (Avaya)	Relay	YES
<b>G.711</b>	NO	NO	NO	NO	NO	NO	Relay

*Note: Avaya option refers to the switch selected during endpoint configuration.*

## Media Services

### Introduction

Media services refers to how call media is handled by the MSX software. Media services provides a simplified interface for the user to configure and use MSX media-handling devices for media routing as well as multiple purposes, such as transcoding, QoS, encryption, etc. These uses can require multiple physical devices including the internal media processor card or an external processing devices such as a transcoder. Media services hides these complexities, giving a view of a single firewall device with all these abilities.

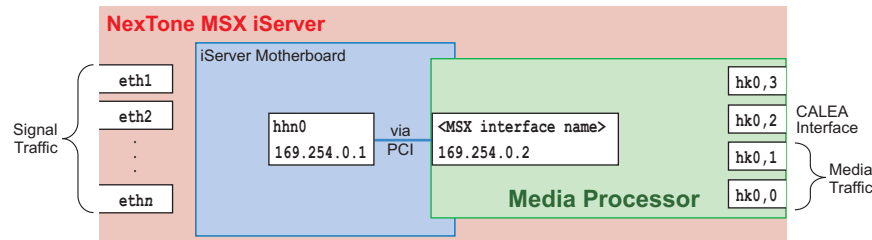
The firewall entity is called the Firewall Control Entity (FCE). The signaling layer of the iServer connects to the FCE, and the FCE connects to its subdevices: e.g., media processor ports, or transcoder.

### About Media Processors

MSX supports intelligent media processor cards for routing call media data. This greatly increases the media handling capacity of each MSX by shifting the primary network traffic processing burden from the main CPU to the CPU on the media processor. MSX supports two media processor cards: NSF-NP which has three interfaces and NSF-NP2 which has four interfaces. NSF-NP2 also supports additional functions such as DTMF generation and translation (see *DTMF Translation and Tone Generation*, on page 331 for more information).

To simplify maintenance, the OS software is loaded at boot time from a file on the host MSX. The figure below illustrates the relationship between the two systems.

**Figure 42. Media Processor Interfaces**



Note: There are three interfaces on the NSF-NP media processor, two of them are used for normal operation: hk0,0 and hk0,1. The hk0,2 interface is used only for Lawful Intercept processing. The NSF-NP2 media processor's first three interfaces are named and used similarly, and it includes a fourth interface, hk0,3, for normal operation.

### **Inter-system communication**

The MSX and the media processor communicate with each other via the fixed, non-routable IP addresses shown in the illustration above. The MSX sees the media processor at 169.254.0.2; the media processor sees the MSX at 169.254.0.1.

### **Media routing support**

The MSX supports only one type of media routing at a time. This can be:

- ♦ An internal media processor card firewall for hardware-based routing
- ♦ NSF for software-based routing

Note: Do not install NSF software on systems that have a media processor card installed.

### **MSX Boot with Media Processor**

When the MSX boots up having as part of its configuration a media processor, it sends a signal to the media processor instructing it to load its OS from a file on the MSX. To confirm that the media processor is running, type the

following command from the MSX command line while logged in as the root or Superuser:

```
rsh 169.254.0.2 ps -eaf | grep 2611
```

If the media processor is running normally, the output from the above command will include at least one line showing `./Control_2611` (usually there will be several).

### **MSX configuration**

Once the media processor is installed, you must configure its interfaces. Depending on the type of card you have installed, three or four media interfaces can be configured on the media processor.

`hk0,0`, `hk0,1` and `hk0,3` are typically configured for media routing. Setting up Media Devices and Media Routing Pools, on page 276 provides instructions for configuring these interfaces for media routing and for use with a transcoding device. To configure the interfaces for media routing, perform the procedures beginning with *Setting up media devices and media routing pools*, on page 297.

The `hk0,2` interface is used exclusively with the Lawful Intercept feature. See “Lawful Intercept”, on page 396 for information about configuring this feature.

## **About NSF**

The NexTone Session Filter (NSF) provides a low-cost, entry-level media routing solution that uses MSX on-board Gigabit Ethernet interfaces. The NSF provides two dedicated Ethernet ports for media routing in addition to the ports for management, redundancy control and signaling. Additional Network Interface Cards (NICs) must be provided for platforms lacking sufficient Ethernet ports. The NSF runs on all current MSX hardware platforms and with version 4.2 or higher MSX software.

Unlike media processor hardware-based solutions, NSF is software-based and does not provide the same level of media throughput rates as hardware-based solutions. Media processor-based solutions **cannot** be implemented on the same platform as the NSF.

Note: See the *NexTone Multiprotocol Session Exchange (MSX) Installation Guide, Release 4.3, Issue 1* for the steps to install and configure NSF.



## Setting up media devices and media routing pools

Media devices are configured in the FCE panel of the **iServerConfiguration** dialog using RSM Console. You then create media routing pools for these devices. Media routing pools are the entities assigned to realms to give a realm its networking identity. The networking definition represented by a media routing pool generates media routing rules for the realm.

The FCE panel contains the three frames and their contents shown in Figure 43, which you use to build media routing pools:

- ♦ **Media device.** A logical identity, representing a device of a certain type, for which you create Vnets and media resource pools.
- ♦ **Media Vnet.** An association of a VLAN ID and a physical interface for a (non-transcoder) media device.
- ♦ **Media resource pool(s).** One or more associations of a Vnet, and an IP address, netmask and port range, for a given (non-transcoder) media device. You can have multiple media resource pools for a given media device that use the same Vnet and have different port ranges, or even IP addresses.
- ♦ **Media routing pools.** A group of one or more media resource pools. A media routing pool can contain one or more resource pools for a single device or, in the case of a transcoder media routing pool, resource pools for two—a firewall device resource pool and a Transcoder resource pool.

Note: The **Internal Interfaces** text field at the top of the FCE page lists the internal interfaces that will not be firewalled. The interfaces in this list typically include the interface used for high-availability communications, and possibly the management interface. A comma-separated list of interface aliases (eth0, eth1, etc.) can be entered manually or they can be defined in the “firewall” section of the nxconfig.pl utility started with the -E option.

### Media Device Types

Each media device is first added to the **Media Devices** frame. Then, the media routing pool(s) for it are created under the device.

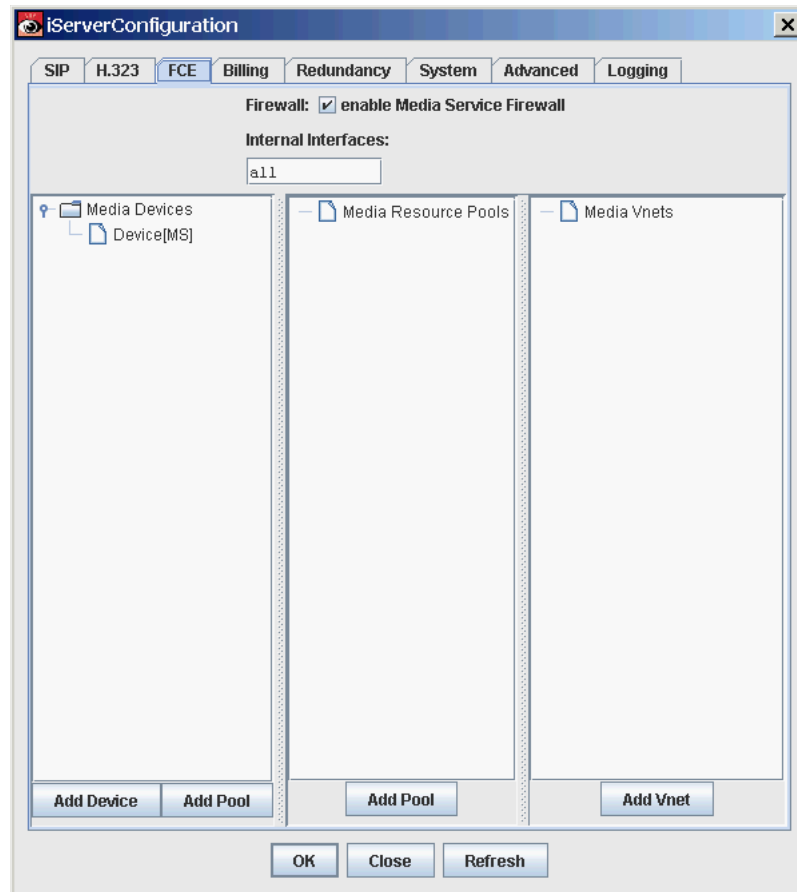
Supported device types include:

- ♦ Media processor card for hardware-based media routing
- ♦ NSF (NexTone session filter) for software-based media routing

- ◆ Transcoder

The first step in the process of creating media routing pools is to create devices.

**Figure 43. iServerConfiguration FCE Page**



### ***About the media routing pools procedures***

The following sections provide procedures for:

- ◆ Creating a media routing pool for media processing
- ◆ Creating a media routing pool for transcoding

The procedure for creating a media routing pool for transcoding is the same as for creating a media routing pool for media processing, except you create and configure resource pools for two devices, a media routing device and a

transcoder device, and then create a media routing pool containing both devices.

The following procedures, through *Create a media routing pool*, on page 304, describe how to create a media routing pool for a single media routing device interface. The procedures starting with *Add a transcoder device*, on page 305, build on these procedures to create a media routing pool for transcoding that includes a media routing device resource pool. To create a transcoder media routing pool, perform all of the following procedures in the indicated order. You will end up with two media routing pools: one containing media routing resources for a media routing device used to send and receive media to/from the transcoder, and one for transcoding that contains media resources for both the transcoder and the media processor card.

#### Summary Steps:

1. Open the iServer configuration FCE page (page 300).
2. Add a media routing device (page 300).
3. Add a Vnet (page 301).
4. Create a media resource pool (page 302).
5. Create a media routing pool (page 304)
6. Add a transcoder device (page 305).
7. Create a media routing pool for transcoding (page 307).

To route transcoded media to multiple realms, create multiple media routing pools from media routing resource pools you create, then configure each realm to use a media routing pool. You only create one transcoder pool per transcoding device, since this pool defines routing information between the iServer and a transcoder device. Currently, the MSX transcoding feature supports only a single transcoding device, so only one transcoder media routing pool should be created. Note that for reasons of security, you should never assign a transcoder media routing pool to a realm.

**Note:** Media Services configuration changes the `mdevices.xml` file stored in the `servercfg` table. Changes to `mdevices.xml` require an MSX restart. When changes that affect the file are made in RSM Console, a message box providing an option to restart the iserver processes will appear. If you decline to restart the processes, you must restart them manually from the MSX command line (by issuing `iserver all stop/iserver all start` commands) before the changes will be

effective. In-process H.323 calls, all incomplete call setups, are dropped during a restart.

### **Open the iServer configuration FCE page**

1. From the iServer main menu, select **iServer > Configure**.  
The iServer Configuration dialog box opens.
2. Click the **FCE** (Firewall Control Entity) tab.  
The **FCE** page shown in Figure 43 appears.  
Because no devices have been configured as yet in our example, no devices appear in the device tree.
3. Select the **enable media Service Firewall** checkbox.
4. Go to Add a media routing device (page 300).

### **Add a media routing device**

1. Select the Media Devices folder in the leftmost pane of the FCE dialog and Click **Add Device**.

The **Add a New Device** dialog box opens.

**Figure 44. Add a New Device Dialog Box**



2. In the **Name** field, type a descriptive name for the media device you are adding.
3. Select a media firewall device from the **Type** drop-down menu. If the iServer has a media processor card installed, select the **hknife** option. If you do not have a media processor card and must use software-based media routing, select the **nsf** option.

4. Click **Set**.

The **Add new device** dialog box closes. The media device you added appears in the device tree.

5. Perform the procedure *Add a Vnet*, on page 301.

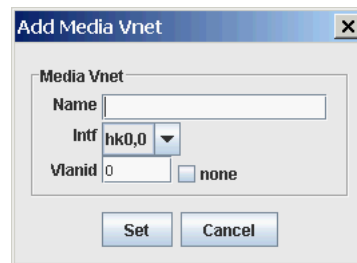
### **Add a Vnet**

In this procedure, you will create a Vnet for the media processor by associating an optional VLAN ID an interface on the media processor card.

1. In the rightmost pane, select the device you just created and click **Add Vnet**

The Add Media Vnet dialog shown in Figure 45 opens.

**Figure 45. Add Media Vnet Dialog**



2. In the **Name** field, give the Vnet a descriptive name. Note that the Vnet will be listed with the selected interface appearing after the name, with the prefix “Vnet”.
3. From the **Intf** pull-down menu, select the physical interface the Vnet applies to.
4. In the **Vlanid** field, choose a VLAN ID to associate with the Vnet. The Vnet will comprise traffic on the indicated interface with the indicated VLAN ID. Note that you can create multiple Vnets that have the same interface, but each must have its own unique VLAN ID. If you don’t want to associate the Vnet with a particular VLAN ID, select **none**.
5. Click **Set**.
6. Routing table entries can optionally be attached media Vnets. To add a media route for the Vnet, select a Vnet, right click, and select **Add Route**

from the context menu. The Add a Media Route dialog shown in Figure 46 appears.

**Figure 46. Add Vnet Media Route Dialog**



7. In the **Dest IP** field, enter a destination IP address for the route (or 0.0.0.0 if you defining the default route). In the **Mask** field, enter an IP mask for the route. In the **Gateway** field, enter the route gateway as required. T, enter as the IP address.
8. Perform the procedure *Create a media resource pool*, on page 302.

### **Create a media resource pool**

In this step, you will create a media resource pool. You can think of a media resource pool as a virtual interface on a physical interface, specified as a Vnet, an IP address and network mask, and a range of ports that can be used on the virtual interface for sending and receiving media. You can have multiple media resource pools for a given device, using the same or different Vnets. If the same Vnet is used, you can assign different IP addresses for each, or use the same IP address. However, if you use the same IP address, the port ranges must not overlap.

1. From the middle pane of the FCE page, select the device for which you are creating a media resource pool.

2. Click **Add Pool**. The Add a New Pool dialog shown in Figure 47 appears.

**Figure 47. Add a New (resource) Pool Dialog**

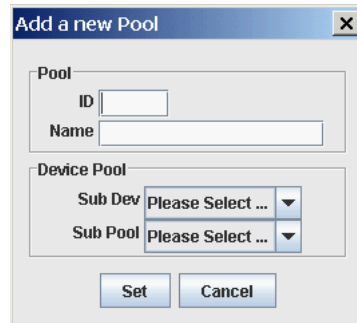
3. In the **ID** field enter an ID number, unique among configured media resource pools, from 1 to 255.
4. In the **Name** field, enter a name descriptive of the pool's purpose, like "Private Network", or "Public Network", and so on. The media resource pool will be listed with the name you enter, prefixed by the pool ID.
5. From the **Vnet** pull-down, select a Vnet created with the previous procedure.
6. In the **IP Address** field, enter an IP address for the resource pool. This binds the IP address to the selected Vnet and the address will be the Realm Media Address (RMA) of the media routing pool that uses this resource pool.
7. In the **Mask** field, enter a network mask for the resource pool.
8. In the **Low Port** and **High Port** fields, create a port range between 11000 and 65535. In **Low Port**, enter the beginning port number for the range. In **High Port**, enter the ending port number for the range.
9. Click **Set**. An entry for the media resource pool appears under the selected device in the middle pane.
10. Perform the procedure *Create a media routing pool*, on page 304.

### Create a media routing pool

In this procedure you will create a media routing pool for the media processor created and configured in the previous procedures.

1. In the leftmost pane, click **Add Pool**. The Add a New Pool dialog appears.

**Figure 48. Add a New (Media Routing) Pool Dialog**

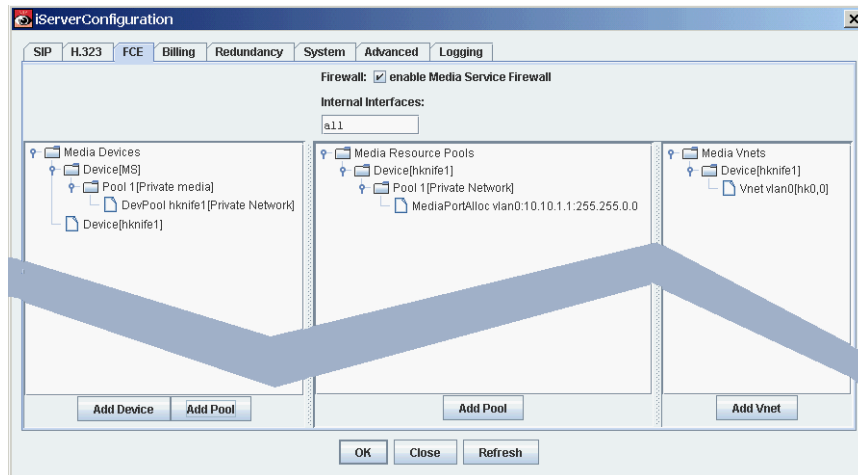
The image shows a Windows-style dialog box titled "Add a new Pool". It has a close button (X) in the top right corner. The dialog is divided into two main sections. The first section, labeled "Pool", contains two text input fields: "ID" and "Name". The second section, labeled "Device Pool", contains two pull-down menus: "Sub Dev" and "Sub Pool", both of which currently display "Please Select ...". At the bottom of the dialog are two buttons: "Set" and "Cancel".

2. In the **ID** field, enter an ID number for the media routing pool, from 1 to 255. The number must be unique among the media routing pools.
3. In the **Name** field, enter a descriptive name for the pool, like "Public Media," or "Transcoding Media", and so on.
4. From the **Sub Dev** pull-down menu, select the device containing the resource you created in the previous procedure (*Create a media resource pool*).
5. From the **Sub Pool** pull-down menu, select a resource pool. There can potentially be multiple pools under the selected device. Select a pool to add to this media routing pool.



6. Click **Set**. The media routing pool appears under the MS device.  
Figure 49 shows the FCE page with the elements of a media processor configured.

**Figure 49. FCE Page with Configured Media Processor Media Routing Pool**



7. Repeat the preceding procedures for any additional (non-transcoder) media devices for which media routing pools are required.
8. If you have a transcoder, perform the procedure *Add a transcoder device*, on page 305. Otherwise, you are finished creating media routing pools.

### **Add a transcoder device**

If you are using the transcoding feature described later in this chapter, you need to create a media routing pool containing resources for the transcoder device and the interface on the media processor that sends/receives media to/from the transcoder device. (For more information on implementing transcoding in MSX, refer to *Configuring MSX for transcoding* on page 320.)

1. Click the **Add Device** button located under the Media Devices (leftmost) pane.  
The Add a New Device dialog opens.

**Figure 50. Add a New Device Dialog Box**

The dialog box titled "Add a new Device" contains the following fields:

- Device** section:
  - Name**: A text input field.
  - Type**: A pull-down menu.
- Control Entity** section:
  - type**: A pull-down menu with "local" selected.
- Forward Entity** section:
  - type**: A pull-down menu with "local" selected.

At the bottom are "Set" and "Cancel" buttons.

2. In the name field, enter a descriptive name for the device, for instance, Transcoder.
3. From the **Type** pull-down, select tcf.  
The Add a New Device dialog expands to show additional configuration fields for the transcoder, as shown in Figure 51.

**Figure 51. Expanded Add a New Device Dialog Box**

The expanded dialog box titled "Add a new Device" contains the following fields:

- Device** section:
  - Name**: A text input field.
  - Type**: A pull-down menu with "tcf" selected.
- Control Entity** section:
  - type**: A pull-down menu with "local" selected.
- Forward Entity** section:
  - type**: A pull-down menu with "pnpcp" selected.
  - ipaddress**: A text input field with "10.10.0.10".
  - mask**: A text input field with "255.255.255.0".
  - port**: A text input field with "2424".
  - Pool Name**: A text input field with "mediant3000".
  - card**: A text input field with "TP-6310".
  - my-ipaddress**: A text input field with "127.0.0.1".

At the bottom are "Set" and "Cancel" buttons.

4. Enter the information in the following fields:
  - ✧ In the **ipaddress field**, enter the IP address of the transcoder. This is the address used by the MSX to send media data to the transcoder.

The addresses in the **my-ipaddress** and **ipaddress** fields must be on the same subnet.

Note: Do not assign the Management IP address to the transcoder. Use a different IP address.

- ✧ In the **mask** field, enter the subnet mask for the iServer/Transcoder subnet (**mask**). Note that the IP address/mask must be on the same subnet as the iServer interface used to communicate with the transcoder.
- ✧ In the **port** field, enter the port number used for MSX-transcoder communication. The default value is 2424, which is the default port for the transcoder protocol. This value should not be changed.
- ✧ The value in the **Pool Name** field is the name that will represent the media resource pool for the transcoder. The default is “mediant3000”. Note that you do not have to create this media resource pool; you only have to give it a name.
- ✧ In the **card** field, specify the card type for this transcoder, TP-6310 or TP-1610. You must use one of these two strings.
- ✧ In the **my-ipaddress** field, enter the IP address of the iServer. This is the IP address the transcoder uses to send transcoded media data to the iServer. This should be an IP on an interface that is lightly use and guaranteed available, such as the Management Interface. The addresses in the **my-ipaddress** and **ipaddress** fields must be on the same subnet.

Note that the value in the **type** field of the Forward Entity section, `tpncp`, is the protocol used by the transcoder to communicate with the iServer and cannot be changed.

5. Click **Set**. The transcoder device is listed under the Media Devices folder.
6. Perform the procedure in *Create a media routing pool for transcoding*, on page 307.

### **Create a media routing pool for transcoding**

So far, you have created one or more resource pools for one or more media processors and implicitly created a resource pool for the transcoder device.

Now you need to include resource pools for both devices in a new transcoding media routing pool.

1. Click **Add Pool**. The Add a New Pool dialog opens.
2. In the **ID** field, enter an ID number for the media routing pool between 1 and 255. The number must be unique among the media routing pools.
3. Enter a descriptive name for the pool in the **Name** field. For this example, it will be named “Transcoding Pool”.
4. From the **Sub Dev** pull-down, select the media processor that will be included in the pool.
5. From the **Sub Pool** pull-down, select the media routing media resource pool to use.
6. Click **Set**. The new Pool, labeled “Transcoding Pool”, appears under the MS device.
7. Select the new “Transcoding Pool” pool, and right click. At the context menu, select **Add DevPool**. The Modify DevPool dialog appears.
8. From the **Sub Dev** pull-down menu, select the name of the transcoder device.
9. From the **Sub Pool** pull-down, select the Resource Pool Name you specified when you created the Transcoder device (see Figure 51).
10. Click **Set**. The new resource pool for the transcoder is added to the “Transcoding Pool” pool.

The transcoder is now configured.

## Reconfiguring devices, pools, and Vnets

Most of the items you create in the FCE page can be reconfigured. To reconfigure an item:

1. Select the item in the appropriate pane of the configuration pane.
2. Right-click the item and select the **Config ...** option from the context menu that appears.

The configuration dialog for the selected item appears. See the appropriate procedure under *Setting up media devices and media routing pools*, on page 297 for specific instructions on using the dialog.

## Deleting devices, pools, and Vnets

Most of the items in the FCE page can be deleted. To delete an item:

1. Select the item to delete from one of the configuration panes.
2. Right-click the item and select the **Delete ...** option from the context menu that appears.
3. If a confirmation message box appears, confirm the deletion.

## Implementing transcoding in MSX

One of the subdevices supported by Media Services in MSX is a transcoder. A transcoder is a digital signal processor (DSP) device that translates between ingress and egress media streams that use various coder-decoders (codecs) or that have different stream characteristics, such as bit rates.

Before you implement transcoding, you should become familiar with the transcoding concepts explained in the following sections:

- ♦ *About transcoding* on page 309
- ♦ *Transcoding network topology* on page 311
- ♦ *The transcoding negotiation process* on page 312

The tasks you need to perform to implement transcoding in MSX are described in these sections:

- ♦ *Transcoding implementation guidelines* on page 315
- ♦ *Configuring MSX for transcoding* on page 320

### ***About transcoding***

Transcoding makes media sessions possible between two devices with incompatible codec requirements. Transcoding provides support for calls traversing networks utilizing various codec types, or the same codec types, but with differing parameters. The iServer implementation of transcoding includes RFC 2833 (DTMF via RTP) and T38 fax support. Other characteristics of the transcoding feature include:

- ♦ SIP to SIP, H.323 to H.323 and IWF calls can be transcoded.
- ♦ Transcoding requires a feature license and a transcoding device.

- ♦ With the currently supported network topology, on an MSX, each transcoded call requires twice the number of media-routed legs.
- ♦ The v4.3 MSX integrates with AudioCodes *Mediant™ 2000* and *Mediant™ 3000* rack-mount units. For information on installing the AudioCodes units, refer to the *NexTone Multiprotocol Session Exchange (MSX) Installation Guide*.

**Important:** Transcoding is supported only with hardware media routing using an internal media processor card. It is not supported with NSF (software) media routing.

Call legs using different codecs, the same type codecs with differing parameters, or different methods to transmit DTMF, require transcoding to interoperate. For example, a call from a conventional packet-switched network gateway supporting only the G.711 $\mu$ -law codec requires transcoding before being passed on to a VoIP gateway *not* supporting the G.711 $\mu$ -law codec. In this case, the DTMF tones carried over the G.711 may also require transcoding to an RFC 2833 format if the egress gateway codec cannot reliably carry DTMF audio (for example, G.723).

Transcoding on the MSX is implemented using dedicated transcoding hardware, letting the MSX control call and session logic while not burdening it with processing-intensive media transcoding.

Transcoding can be invoked dynamically upon receiving a certain media-specific signaling error, or it can be invoked using static policies assigned to source and destination endpoints. In the static scenario, an ingress offer is modified as it traverses the MSX in consecutive stages until the egress offer is acceptable to the egress endpoint. (See *The transcoding negotiation process*, on page 312.)

The MSX makes transcoding decisions for each call, attempting to match the ingress endpoint codec capabilities with those supported by the egress endpoint. Where no direct match is available, the ingress media stream is transcoded to a supported egress endpoint codec.

Endpoint codec characteristics are statically provisioned using codec profiles. Codec profiles can be created either from RSM Console or using CLI. One codec profile can be associated with multiple endpoints.

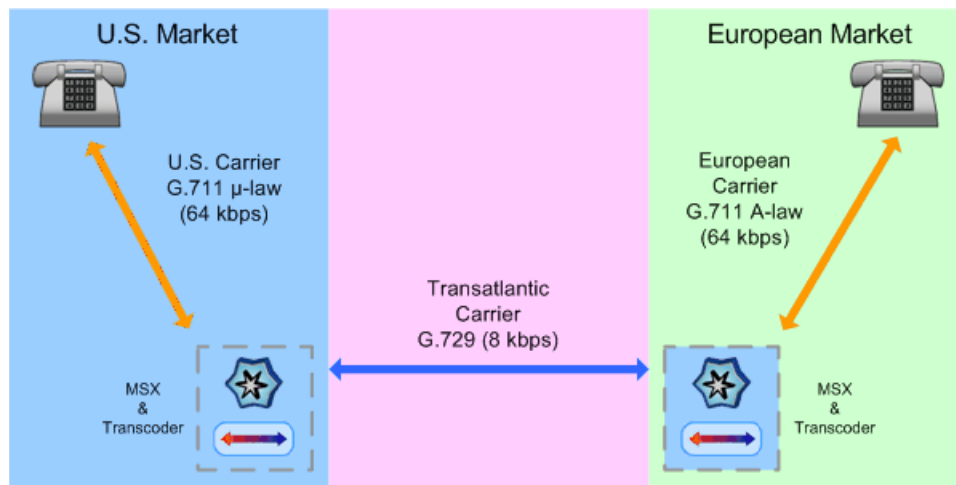
## Transcoding network topology

You can configure the transcoder to route all media to and from it via the MSX. This provides maximum security but also loads the MSX with additional media streams for the same number of calls.

### Preferred, prohibited, and non-preferred codecs

Figure 52 illustrates three different codecs used in one call path.

**Figure 52. Codecs in One Call Path**



This is one common application of transcoding used in minimizing bandwidth for the transatlantic leg, and utilizing commonly used codecs for the U.S and Europe.

On the MSX, codec selection preferences and prohibitions are collected in a named entity called a “codec profile.” Codec profiles define the codec preferences and prohibitions for that endpoint. Every endpoint participating in transcoding services controlled by the MSX must specify a codec profile to use.

Each codec profile is comprised of two lists:

- ♦ The Preferred Codecs list is a prioritized list of codecs that the administrator prefers the endpoint to use. Codecs on this list appear in order of preference.
- ♦ The Prohibited Codecs list is a list of codecs that the administrator *will not allow* on that connection. They are stripped from offers (that is, from

SIP INVITES and H.323 setups) before they are forwarded to the corresponding ingress or egress endpoint.

### ***The implied non-preferred codec list***

It is possible for a codec to be on neither the Preferred Codecs list nor the Prohibited Codecs list. This effectively places it on an “implied” third list of codecs that can still be used under certain conditions if the endpoints negotiate to use it.

This implied list is useful, for example, if a new or previously unknown codec comes into use. In such a case, the offer is forwarded with the new codec not stripped out (because it is not on the Prohibited list; refer to *The transcoding negotiation process*, on page 312). Implied-supported codecs are placed in priority after the ones in the preferred list. Because this implied list is available, implementing a new codec may not require reworking existing codec profiles.

There may be certain situations, however, under which you wish to limit available codecs to *only* those on the Preferred list. To accomplish this, you specify the all option in the Prohibited list.

### ***About endpoints negotiating codecs***

During the course of call setup and session parameter negotiation, codec types and settings are proposed by the ingress and egress endpoints, then a scheme is settled upon that both endpoints agree to support. Exactly how the selection is made depends on how the endpoints negotiate session particulars, and is beyond the scope of this document. However, the codec selection process carried out by the MSX is designed to increase the *probability* of negotiating a successful session.

### ***The transcoding negotiation process***

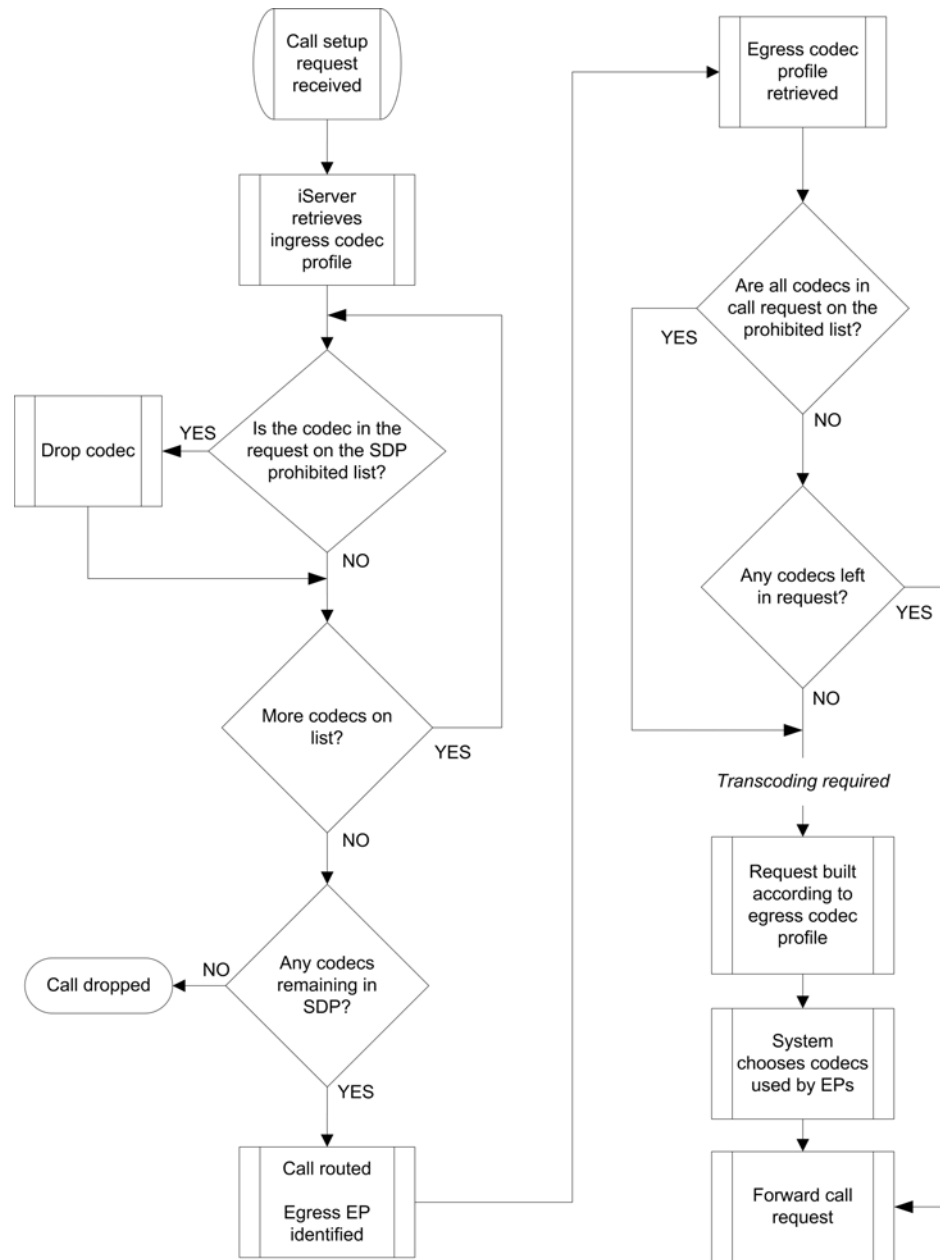
1. A call setup request is received.
2. The MSX retrieves the codec profile for the *ingress* endpoint.
3. Any codecs proposed in the offer that are found on the prohibited list for the *ingress* endpoint are removed from the offer. Codecs that remain in the offer are preferred (or at least not prohibited) by the ingress endpoint. Note that any answer subsequently returned from the *egress* endpoint that contains one of the codecs on this list *will not* be in violation of the *ingress* endpoint’s codec policy.



4. The codecs listed in the offer are re-ordered. The first one found in a first-to-last search that is also on the preferred list is the first one offered.
5. Call routing logic determines the call's egress endpoint.
6. The MSX retrieves the *egress* endpoint's codec profile and checks it against the list of codecs in the offer.
7. The process continues, according to this table:

If...	Then...
the endpoints cannot negotiate a session	the call is not completed.
the egress endpoint has no codec profile assigned to it	the offer is forwarded as-is, and the call cannot be transcoded.
at least one codec in the offer is on either the egress endpoint profile's <i>preferred</i> list or its implied list	the call goes through without transcoding because none is required.
no codec in the offer is on the profile's prohibited list	the call goes through transcoded.

The flow chart in Figure 53 illustrates the process.

**Figure 53. The Transcoding Negotiation Process**

## Transcoding implementation guidelines

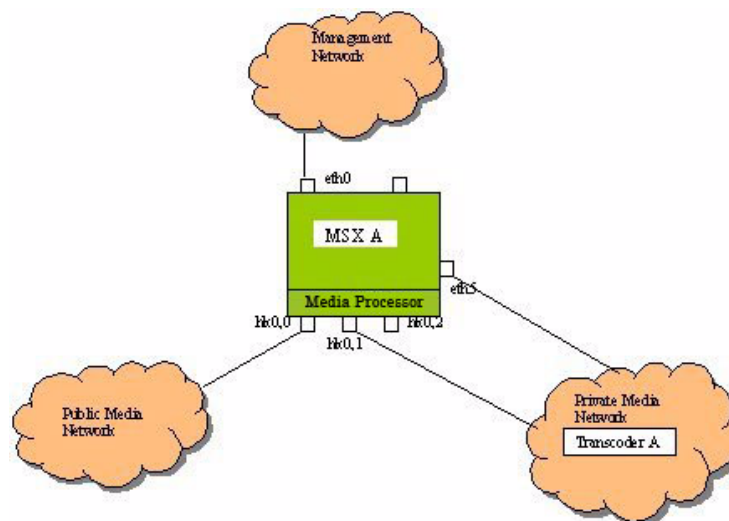
The information in the following sections provides you with guidelines for implementing transcoding in MSX:

- ♦ *Configuring standalone and redundant systems* on page 315
- ♦ *System behavior during operation* on page 317
- ♦ *Troubleshooting transcoder problems* on page 319
- ♦ *Limitations* on page 319

### Configuring standalone and redundant systems

In a standalone configuration, MSX uses eth5 for signaling control. Figure 54 shows a typical transcoder configuration for a standalone MSX system.

**Figure 54. Transcoder configuration for a standalone MSX system**



With a redundant configuration, eth5 on the master and slave system must have different IP addresses, since master and slave systems communicate with the transcoder in active/standby mode.

Figure 55 shows a typical transcoder configuration with redundant MSX systems.

**Figure 55. Transcoder configuration for a redundant MSX system**

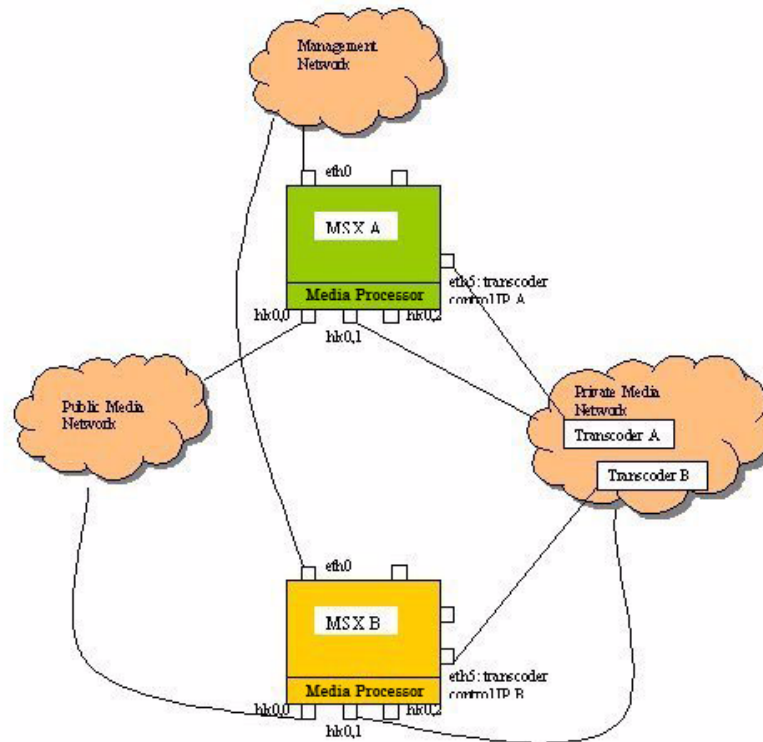


Figure 55 shows two transcoders, each attached to one of the MSX systems. Separate mdevices.xml files are required for each of the MSX systems in the redundant pair and should be configured as follows:

1. The Realm Media Addresses (RMA) required for signaling should be configured with the same IP address.
2. The AudioCodes device configured in each of the mdevices.xml files should specify the IP addresses of the respective AudioCodes boxes. The AudioCodes device should also specify the eth5 IP address of the MSX with which it connects to the AudioCodes unit.

3. The media processor should be configured with an additional media routing pool. This pool is used to allocate ports for redirects to the AudioCodes unit and should be in the same pool group as the AudioCodes unit. The descriptive name you assign to this pool group should match the pool group you created for the AudioCodes unit. See *Create a media routing pool* on page 304.

**Important:** The RMA configured for the transcoder pool group must be unique for the active and standby MSXs.

### ***System behavior during operation***

System behavior during operation concerns what happens when the transcoder fails. When MSX is operating, it detects transcoder failure by sending and listening for keep-alive messages. When keep alive messages fail to arrive within five seconds and there is no traffic on the control connection between a MSX and the transcoder, failure detection is triggered.

The `restart-iserver-on-transcoder-fail` system configuration parameter controls the behavior when the system detects transcoder failure. To configure MSX to perform a switchover when a failure is detected, enter the following command:

```
Command: nxconfig.pl -e restart-iserver-on-transcoder-fail -v 1
```

By default, `restart-iserver-on-transcoder-fail` is 0.

Table 48 describes system behavior on both redundant and stand-alone MSX systems when the system detects transcoder failure.

**Table 48. System behavior when a transcoder failure is detected during operation**

	<b>restart-iserver-on-transcoder-fail = 1</b>	<b>restart-iserver-on-transcoder-fail = 0</b>
<b>Redundant MSX configuration</b>	<p>The active MSX initiates a switchover, switches over all active calls, and drops transient calls.</p> <p>The slave MSX attempts to reconnect to the transcoder.</p>	MSX attempts to reconnect to the transcoder.
<b>Stand-alone MSX configuration</b>	MSX attempts to reconnect to the transcoder.	MSX attempts to reconnect to the transcoder.

#### **System behavior during startup**

System behavior during startup concerns what happens when the transcoder cannot be reached. Table 49 describes this behavior for redundant and stand-alone MSX configurations.

**Table 49. System behavior when transcoder cannot be reached during startup**

<b>Redundant MSX configuration</b>	<b>Stand-alone MSX configuration</b>
<p><b>Master MSX</b></p> <ul style="list-style-type: none"> <li>Every 30 seconds, the active MSX tries to reconnect. It does this up to 5 times before it initiates a switchover.</li> </ul> <p><b>Slave MSX</b></p> <ul style="list-style-type: none"> <li>MSX attempts to reconnect to the transcoder.</li> </ul>	MSX attempts to reconnect to the transcoder.

### Troubleshooting transcoder problems

The following tips may help in the event that you have problems with the AudioCodes transcoder.

Problem	Action
Call failure.	<p>Check the redirects (<code>./statclient -h dumpredir</code>) for a transcoded call. You will see four redirects, if working correctly.</p> <pre> 2      0    UDP    010.013.001.057:1030 010.013.001.056:1032    172.016.001.200:16598 1      0    0x0020 0x1833 00:00:50:11:6A:25 00:0F:8F:A2:98:00  0      0    UDP    010.013.001.055:1032 010.013.001.057:1032    010.013.001.150:4030 2      0    0x0064 0x1833 00:00:50:11:6A:26 00:90:8F:05:32:B9  2      0    UDP    010.013.001.057:1032 010.013.001.055:1032    172.016.001.189:16582 0      0    0x001f 0x1833 00:00:50:11:6A:24 00:0F:8F:A2:98:00  1      0    UDP    010.013.001.056:1032 010.013.001.057:1030    010.013.001.150:4020 2      0    0x0064 0x1c33 00:00:50:11:6A:26 00:90:8F:05:32:B9 </pre>
Call failure due to the media device file (mdevices.xml) being incorrect.	<p>Check to see if the correct device type and fields appear in the mdevices.xml file:</p> <pre> transcoder-ip subnet my-ip device-specific </pre>

### Limitations

When you implement transcoding for MSX, be aware of the following limitations.

- ♦ Fax transcoding (T.38 to g.711x or g.711x to T.38) T.38 on H.323 leg accepts only 14.4 kbps, regardless of the original voice codec configured.
- ♦ The transition from fax call to voice call may not work in all cases, since MSX currently does not have the capability to fall back to a voice call after a fax call.
- ♦ Fax calls could be lost during a failover.

- ♦ Fax calls may fail if signaling messages arrive before fax-tones.

## ***Configuring MSX for transcoding***

In addition to adding a transcoder device (described in the section *Add a transcoder device* on page 305) and creating a media pool for transcoding (described in the section *Create a media routing pool for transcoding* on page 307), there are other configuration tasks that also must be performed. These tasks are described in the following sections:

- ♦ *Creating codec profiles* on page 320
- ♦ *Configuring endpoints and iEdge groups* on page 330

### ***Creating codec profiles***

Codec profiles may be created using either RSM Console or CLI. The following sections show both methods. In RSM Console, adding, deleting, and modifying codec profiles are all done in the Codec Profiles window. This window can also be accessed from the Database window by selecting **Edit > Add > Endpoint > Provision an Endpoint**.



## USING RSM CONSOLE

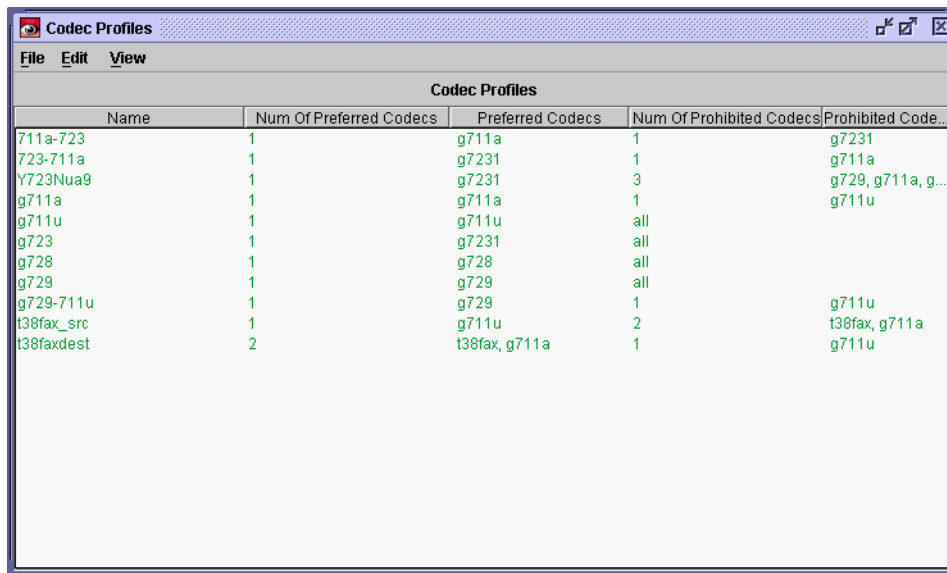
### Adding a codec profile

To add a new codec profile:

1. From the iServer main menu, select **Utilities > CodecProfile**.

The **Codec Profiles** window opens.

**Figure 56. Codec Profiles Window**



The screenshot shows a window titled "Codec Profiles" with a menu bar (File, Edit, View) and a table of codec profiles. The table has five columns: Name, Num Of Preferred Codecs, Preferred Codecs, Num Of Prohibited Codecs, and Prohibited Codecs. The data is as follows:

Name	Num Of Preferred Codecs	Preferred Codecs	Num Of Prohibited Codecs	Prohibited Codecs
711a-723	1	g711a	1	g7231
723-711a	1	g7231	1	g711a
Y723Nua9	1	g7231	3	g729, g711a, g...
g711a	1	g711a	1	g711u
g711u	1	g711u	all	
g723	1	g7231	all	
g728	1	g728	all	
g729	1	g729	all	
g729-711u	1	g729	1	g711u
t38fax_src	1	g711u	2	t38fax, g711a
t38faxdest	2	t38fax, g711a	1	g711u

Previously defined profiles appear in the window.

2. From the Codec Profiles dialog box, select **Edit > Add**.  
The **Add Codec Profile** dialog box opens.

**Figure 57. Add Codec Profile Dialog Box**

Partition: **admin**

Name:

Preferred Codec:

☐ all ☐ none ☒ customize

**g711u**

Prohibited Codec:

☐ all ☐ none ☒ customize

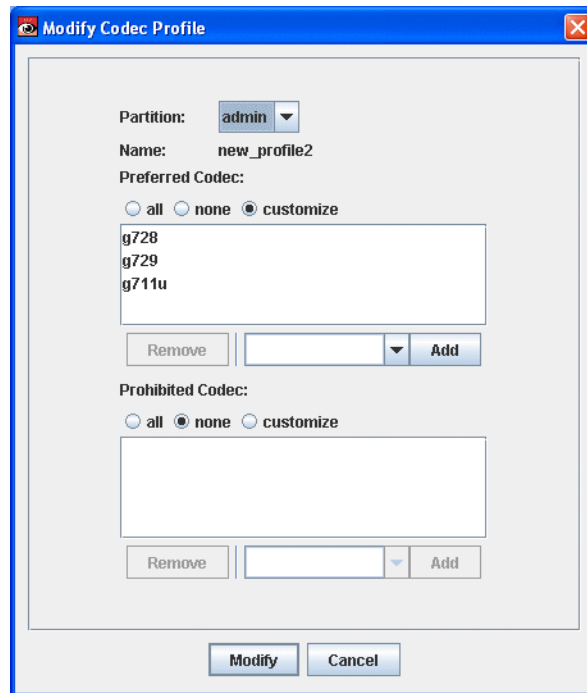
**g711u**

3. In the drop-down list, select the partition to which you want to add the codec profile.
4. In the **Name** box, type a name for the new codec profile.
5. Select **Customize** from the **Preferred Codec** options.
6. Choose the codecs you want to support by selecting the codecs from the drop-down menu in the preferred codecs section of the dialog box, then

click the **Add** button located next to the drop-down menu after each selection.

The codecs you added appear in the list of preferred codecs.

**Figure 58. Preferred Codecs List**



7. If you want to prohibit certain codecs from being used, go to the next step. Otherwise, go to Step 11.
8. If you want to prohibit all codecs except those on the Preferred list, choose the **all** option, and proceed to Step 11.
9. Select the **Customize** option under **Prohibited Codec**.
10. Choose the codecs you do not want to support by selecting the codecs from the drop-down menu in the prohibited codecs section of the dialog box, then click the **Add** button located next to the drop-down menu after each selection.

The codecs you chose not to support appear in the prohibited codecs list.

**Figure 59. Prohibited Codecs list**

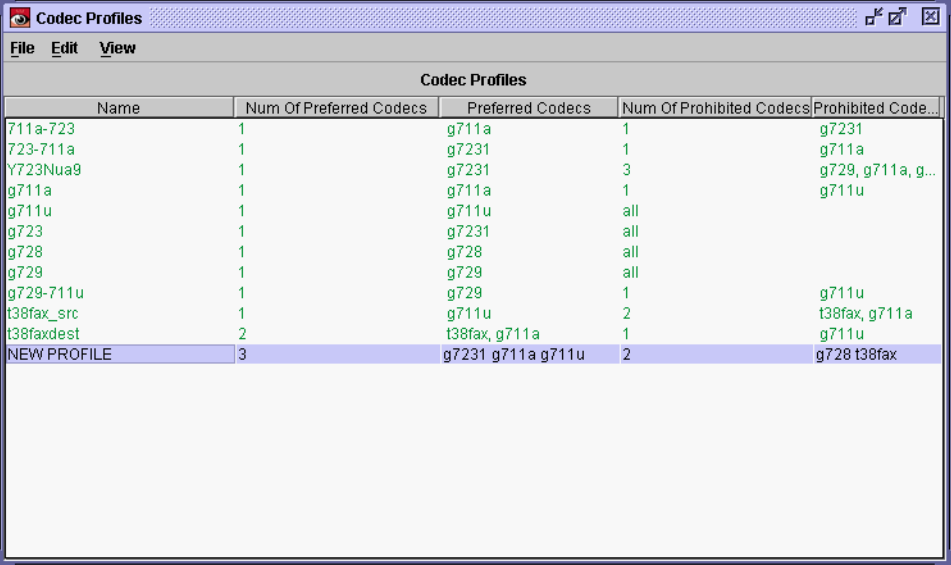
The screenshot shows a 'Modify Codec Profile' dialog box. It has a title bar with a red 'X' button. The dialog contains the following fields and controls:

- Partition:** A dropdown menu showing 'admin'.
- Name:** A text field containing 'new\_profile2'.
- Preferred Codec:** Radio buttons for 'all', 'none', and 'customize' (selected). Below is a list box containing 'g728', 'g729', and 'g711u'. At the bottom of this section are 'Remove', an empty dropdown, and 'Add' buttons.
- Prohibited Codec:** Radio buttons for 'all', 'none', and 'customize' (selected). Below is a list box containing 'g7231' and 'g711a'. At the bottom of this section are 'Remove', an empty dropdown, and 'Add' buttons.
- At the very bottom are 'Modify' and 'Cancel' buttons.

11. When you finish building the profile, click the **Add** button at the bottom of the Add Codec Profile dialog box.

The **Add Codec Profile** dialog box closes. The new profile is listed in the Codec Profiles window.

**Figure 60. New Codec Profile Added**



Codec Profiles				
Name	Num Of Preferred Codecs	Preferred Codecs	Num Of Prohibited Codecs	Prohibited Code...
711a-723	1	g711a	1	g7231
723-711a	1	g7231	1	g711a
Y723Nua9	1	g7231	3	g729, g711a, g...
g711a	1	g711a	1	g711u
g711u	1	g711u	all	
g723	1	g7231	all	
g728	1	g728	all	
g729	1	g729	all	
g729-711u	1	g729	1	g711u
t38fax_src	1	g711u	2	t38fax, g711a
t38faxdest	2	t38fax, g711a	1	g711u
NEW PROFILE	3	g7231 g711a g711u	2	g728 t38fax

### Changing a codec profile

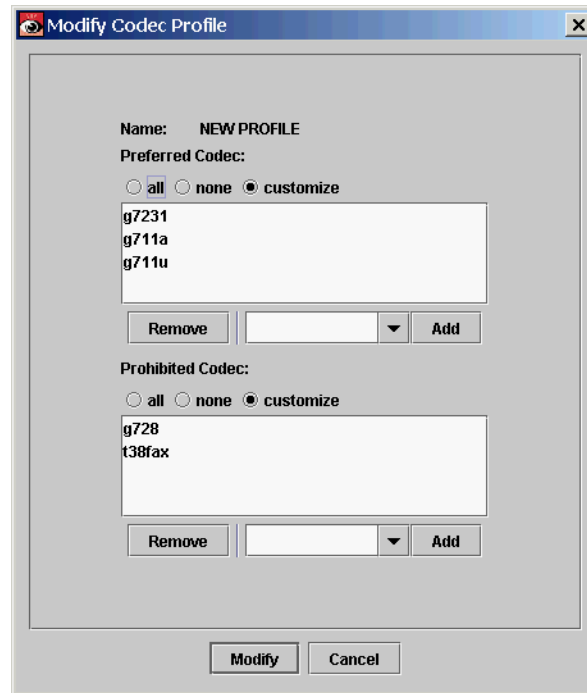
To change a codec profile:

1. From the iServer main menu, select **Utilities > CodecProfile**.  
The **Codec Profiles** window opens.

2. From the list of codec profiles, double-click the profile you want to change, or right-click the profile, then select **Modify**.

The **Modify Codec Profile** dialog box opens.

**Figure 61. Modify Codec Profile Dialog Box**



3. Perform any of the following actions to modify the codec profile:
  - ✧ *To add a preferred codec*, select the codec from the drop-down menu in the preferred codecs section of the dialog box, then click the **Add** button next to the drop-down menu.
  - ✧ *To delete a preferred codec*, select the codec from the drop-down menu in the preferred codecs section of the dialog box, then click the **Remove** button next to the drop-down menu.
  - ✧ *To add a prohibited codec*, select the codec from the drop-down menu in the prohibited codecs section of the dialog box, then click the **Add** button next to the drop-down menu.
  - ✧ *To delete a prohibited codec*, select the codec from the drop-down menu in the prohibited codecs section of the dialog box, then click the **Remove** button next to the drop-down menu.

4. When you finish making changes, click **Modify**.

The system makes the changes in the profile, and the **Modify Codec Profile** dialog box closes.

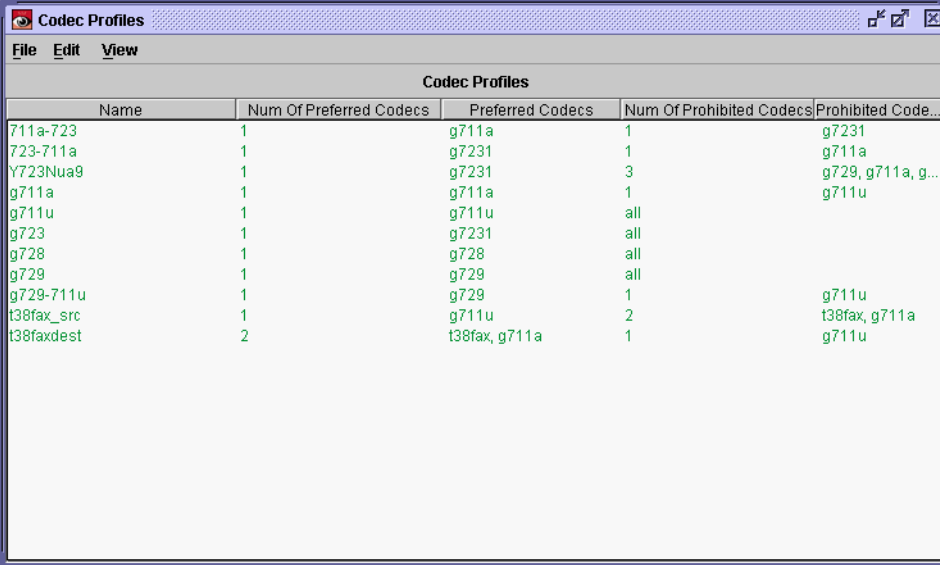
### **Deleting a codec profile**

To delete a codec profile:

1. From the iServer main menu, select **Utilities > CodecProfile**.

The **Codec Profiles** window opens.

**Figure 62. Codec Profiles Window**



Codec Profiles				
Name	Num Of Preferred Codecs	Preferred Codecs	Num Of Prohibited Codecs	Prohibited Codecs
711a-723	1	g711a	1	g7231
723-711a	1	g7231	1	g711a
Y723Nua9	1	g7231	3	g729, g711a, g...
g711a	1	g711a	1	g711u
g711u	1	g711u	all	
g723	1	g7231	all	
g728	1	g728	all	
g729	1	g729	all	
g729-711u	1	g729	1	g711u
t38fax_src	1	g711u	2	t38fax, g711a
t38faxdest	2	t38fax, g711a	1	g711u

2. Right-click the profile you want to delete and select **Delete**.

The system asks you to confirm that you want to delete the selected profile.

3. Click **Yes**.

The system deletes the codec profile and removes it from the Codec Profiles list.

### **Assigning a codec profile to an endpoint**

Once codec profiles are defined, they can be assigned to endpoint uPorts. There are two transcoding configuration settings that are defined for a uPort: A setting that enables or disables transcoding for the uPort, and a setting that defines the codec profile for the uPort.

To enable transcoding and assign a codec profile to an endpoint uPort:

1. Select **iServer > Database > Show** from the RSM Console/RSM Console main window.  
The Database screen appears.
2. Expand the database tree to display the uPorts of the endpoint you want to configure.
3. Double-click the uPort you want to configure. The Phone page of the uPort configuration screen appears.
4. Click the **Enable transcoding** check box.
5. From the Codec Profile drop-down menu, select the name of the codec profile to use for the uPort.

The Phone page will appear similar to the one shown. In this illustration, transcoding is enabled and the codec profile named “NEW PROFILE” is assigned to the uPort.

**Figure 63. EndPoint uPort with Transcoding Enabled**

Modify Media Server: mediasrv/1

Phone Advanced User Info Protocol Calls Media/INFO Config

Partition: admin

Device Type: Media Server

Registration ID: mediasrv

Port Number: 0

IP Address:

Extension:

Phone Number:

VPN Phone Number:

Calling Plan: cp\_4444

Diversion Calling Plan:

Realm: realm3

IEdge Group: 1

☒ Enable transcoding

Codec Profile: new\_profile



Note that if the **Enable transcoding** check box is cleared but a codec profile is selected in the **Codec Profile** menu, only the prohibited codecs will have any meaning.

If the **Transcoding enabled** check box is selected but no codec profile is selected, transcoding will be allowed when the endpoint is the caller, provided the destination has a codec profile. If a destination endpoint has no codec profile, then transcoding will not be activated for the call.

## USING CLI

You can work with profiles using CLI. These CLI commands are supported:

Task	Command	Options/Notes
Create a new codec profile	<code>cli cdc add &lt;codec-profile-name&gt;</code>	31 characters, maximum
Delete an existing codec profile	<code>cli cdc delete &lt;codec-profile-name&gt;</code>	
Edit existing codec profile characteristics	<code>cli cdc edit &lt;codec-profile-name&gt;</code> [prefer <u>codec1,codec2,...</u> ] [prohibit <u>codec1,codec2,...</u> ]	A list of valid codecs can be obtained by entering just <code>cli cdc edit</code> . Enter the codecs in priority order, as a list separated by commas but no spaces.
Show all codec profiles defined	<code>cli cdc list</code>	The command gives details of defined profiles.
Show codec profiles currently in the iServer's active cache	<code>cli cdc cache</code>	
Assign a codec profile to an endpoint	<code>cli iedge edit regid uport cdcprfl &lt;codec-profile-name&gt;</code>	To remove a profile from an endpoint, use " " (i.e., two contiguous double-quote marks) for <u>name</u> .
Turn on transcoding for an endpoint	<code>cli iedge edit regid uport transcode [0/1]</code>	0=Off; 1=On

If an invalid codec name is entered in a list of codecs (either preferred or prohibited), that codec name is ignored. If the list consisted of only that one codec, command execution leaves the list empty for that profile and list type. Therefore, if you get an error message saying you have entered an invalid codec, re-enter the command unless you *want* that list to have no codecs in it.

### Configuring endpoints and iEdge groups

Table 50 lists the CLI commands used to configure endpoints and iEdge groups for transcoding.

The MSX transcoding solution can handle real-time fax transcoding. The `not38support` (no T.38 support) iEdge flag indicates the fax capability of an endpoint— either T.38 or fax pass-through. For a transcoded call, you can configure the following combinations of fax capability:

- ✦ Both endpoints are T.38 capable
- ✦ Both endpoints are not T.38 capable (endpoints support only fax pass-through)
- ✦ One endpoint is T.38 and the other endpoint is fax pass-through

If an endpoint is not capable of supporting T.38, MSX assumes the endpoint is capable of fax pass-through. The codec that is configured on an endpoint that supports transcoding must either be g.711a-Law or g.711u-Law and must match the pass-through codec that is configured on the gateway

Values for SIP and H.323 endpoints can be provisioned only using CLI..

**Table 50. CLI commands used to configure endpoints and iEdge groups for transcoding**

Task	Command	Options/Notes
Set the use of RFC2833 telephone event	<pre>cli iedge edit <u>regid</u> <u>uport</u> 2833capable {yes no}</pre> <pre>cli iedge edit <u>regid</u> <u>uport</u> 2833capable unknown {yes no}</pre>	<p>Yes sets the RFC2833 telephone event to required.</p> <p>Yes sets the RFC2833 telephone event to unknown.</p>
Set a required payload type (number) for an endpoint when it receives RFC2833 information	<pre>cli iedge edit <u>regid</u> <u>uport</u> pt2833 <u>integer</u></pre>	Valid integer values are 96-127.
Configure an endpoint for T.38 capability	<pre>cli iedge edit &lt;reg-id&gt; &lt;uport&gt; not38support disable</pre>	Sets the endpoint for T.38. Disable is the default setting.
Configure an endpoint for fax pass-through capability	<pre>cli iedge edit &lt;reg-id&gt; &lt;uport&gt; not38support enable</pre>	Sets the endpoint for fax pass-through.

## DTMF Translation and Tone Generation

DTMF tones are used in voice mail and automated support applications. DTMF signaling is performed using a variety of protocols, conventions, and standards. The MSX accepts incoming DTMF events in a number of formats and can translate these events into other formats depending on the type of input and the configuration you provide for destination endpoints.

The MSX processes incoming messages with DTMF tones differently according to whether the messages are received as out-of-band (signaling) or in-band (media). When the input is out-of-band, it also bases its processing on whether the messages are SIP or H.323. SIP messages can be either INFO or NOTIFY. H.323 messages can be H.245 UII, H.245 alpha, or Q.931 INFO.

The type of input message and the configuration of the endpoint receiving the messages determines what the outgoing leg contains. The following table summarizes the types of conversions the MSX supports. “Yes” means the MSX performs the conversion between the ingress leg and the egress leg. “Relay” means the MSX passes the message unchanged. “IWF Relay” means the MSX passes the message through its Interworking Functions services. “No” means the conversion is not currently supported.

**Table 51. DTMF Translation Matrix**

Egress Leg	INFO	NOTIFY	H.323 UII(S)	H.323 UII(A)	Q.931	RFC2833	G.711
Ingress Leg							
INFO	Relay	YES	IWF Relay	IWF Relay	IWF Relay (Avaya)	YES	YES
NOTIFY	YES	Relay	YES	NO	YES	YES	YES
H.323 UII(S)	IWF Relay	YES	Relay	Relay	YES (Avaya)	YES	YES
H.323 UII(A)	IWF Relay	YES	YES	Relay	YES (Avaya)	YES	YES
Q.931	IWF Relay (Avaya)	YES	YES	YES	Relay (Avaya)	YES	YES
RFC2833	YES	YES	YES	YES	YES (Avaya)	Relay	YES
G.711	NO	NO	NO	NO	NO	NO	Relay

Note: Avaya option refers to the switch selected during endpoint configuration

## Configuring DTMF processing

The following procedures describe how to enable and configure DTMF translation, with a section on what you will need before running this feature:

### Prerequisites

DTMF signaling translation and tone generation processing requires:

- ✧ License for DTMF\_GENERATE and DTMF\_DETECT
- ✧ You will need an NSF-NP2 media processor card to detect and generate DTMF tones.

### Enabling DTMF Event/Message Handling

To enable DTMF event handling, use the following global command:

```
nxconfig.pl -e enabledtmfxlate -v <0|1>
```

*usage:*

valid values: 1 (enable), 0 (disable)

default value: 0 indicating that DTMF event/message handling is disabled

### Configuring Endpoints for DTMF Event/Message Handling

When you configure an endpoint for DTMF you specify whether you want it to receive DTMF as in-band (media) or out-of-band (signaling) messages. If the endpoint will receive SIP out-of-band messages, then you must also specify whether it should receive NOTIFY or INFO messages.

### *Configuring DTMF In-Band/Out-of-Band for an Endpoint*

The following command is available at the MSX command-line interface to configure whether endpoints will receive DTMF as signaling messages (out-of-band) or in the media stream (in-band):

```
cli iedge edit egressdtmf <default|out-of-band|in-band>
```

### ***Configuring DTMF SIP INFO/NOTIFY for an Endpoint***

The following command is available at the MSX command-line interface to determine whether the endpoint will receive SIP NOTIFY or INFO messages for DTMF events:

```
cli iedge edit sipdtmf <default|info|notify>
```

Note: If you want to relay DTMF straight through from incoming to outgoing endpoints, use the default value for both commands.

### **Configuring DTMF INFO Duration**

DTMF INFO duration can be globally configured using *nxconfig* to control the signal duration placed in outgoing DTMF messages generated in NOTIFY- or media-to-INFO conversions. In simple INFO-to-INFO DTMF relay the incoming event duration is passed through unchanged. The default value for DTMF INFO duration is 200 milliseconds.

Use the following command to set DTMF INFO duration:

```
nxconfig.pl -e sipinfodtmfduration -v <value in milliseconds>
```

### ***Configuration examples***

The following examples illustrate the processing that occurs based on the type of DTMF input and the receiving endpoint's configuration settings:

- ♦ If a SIP INFO message is received, and you configure the receiving endpoint for out-of-band and the INFO message type, the message through without further processing.
- ♦ If a SIP INFO message is received, and you configure the receiving endpoint for out-of-band and the NOTIFY message type, MSX converts the INFO message to NOTIFY.
- ♦ If an RFC 2833 event is received and you configure the receiving endpoint as in-band or default, the endpoint receives an RFC 2833 event through the media processor.
- ♦ If an RFC 2833 events is received and the receiving endpoint is configured for out-of-band and the INFO message type, MSX converts the event to a SIP INFO message.

## Firewall Control Entity (FCE)

*Note: Access to FCE requires a feature license. See Chapter 5, MSX Licenses on page 100 for details on iServer licenses and procedures to obtain them.*

A network firewall prevents unauthorized access to the equipment, customers, and services provided by the network. The MSX uses two classes of firewalls: one for signaling interfaces and one for media. The media firewall, described in this chapter, is defined as a subsystem called a Firewall Control Entity, FCE. The MSX signaling firewall is described in Chapter 17, “Signaling Firewall Operations,” on page 339.

### Supported FCE types

Media firewalling is accomplished using one of two mutually exclusive methods and is configured in the RSM Console FCE server configuration tab:

- The NexTone Session Filter (NSF) software.
- The built-in media processor card.

### Configuring the MSX for FCE support

To use the FCE feature, Media Services (MS) must be enabled. This can be accomplished using the `nxconfig.pl` utility or using the RSM Console FCE configuration page.

#### **Media processor support**

The built-in media processor card supports intelligent media processing. This greatly increases the media handling capacity of the MSX. If your MSX has a media processor card, it was already installed by NexTone. This card is not customer-installable.

#### **NSF support**

The MSX can be configured to filter and process media traffic without the assistance of a media processor by installing and configuring NSF software.

However, this limits the volume of call setups and simultaneous media-routed calls that the MSX can support.

**Caution:** *Do not install NSF software on systems equipped with a media processor card.*

### Configuring FCE in the servercfg table

To use an FCE media services must be enabled. You can enable the Media Service firewall using RSM Console or the `nxconfig.pl` utility. To configure the FCE feature using `nxconfig.pl`:

1. Log into the MSX and enter:

```
nxconfig.pl -e fwname
```

A menu of valid values for the attribute appears:

```
a) => none
```

```
b) => MS
```

```
q) => quit
```

```
Valid values are displayed above
```

```
Select your choice:
```

2. Type `b` to enable the firewall control entity, or `a` to disable it.
3. Restart the MSX by entering:

```
iserver all stop
```

```
iserver all start
```

**Note:** *In-process H.323 calls, all incomplete call setups, are dropped during a restart.*

## Media routing

Media routing via the FCE forces media through a central entity known as a *media router*, thereby providing isolation of media between two or more endpoints. Media routing forces media traffic through the firewalled MSX, so that the endpoint network addresses are hidden from one another; each endpoint sends traffic only to the firewall address, and does not know the address of the endpoint at the other end of the vport.

Media routing provides isolation between the two legs of the call and hides the topologies of the networks connecting to that common point. The FCE performs the media routing functions. Without media routing, when a call is

established between two endpoints outside the firewall, the media traffic flows directly between the endpoints, and not through the firewall.

**Note:** Both endpoints must support media routing for this “address hiding” functionality to work. If the destination endpoint of a call does not have media routing enabled, it will still receive media traffic directly from the source of the call.

Figure 64 illustrates the part each MSX interface plays in processing a media-routed call.

**Figure 64. Media Routing Interfaces**

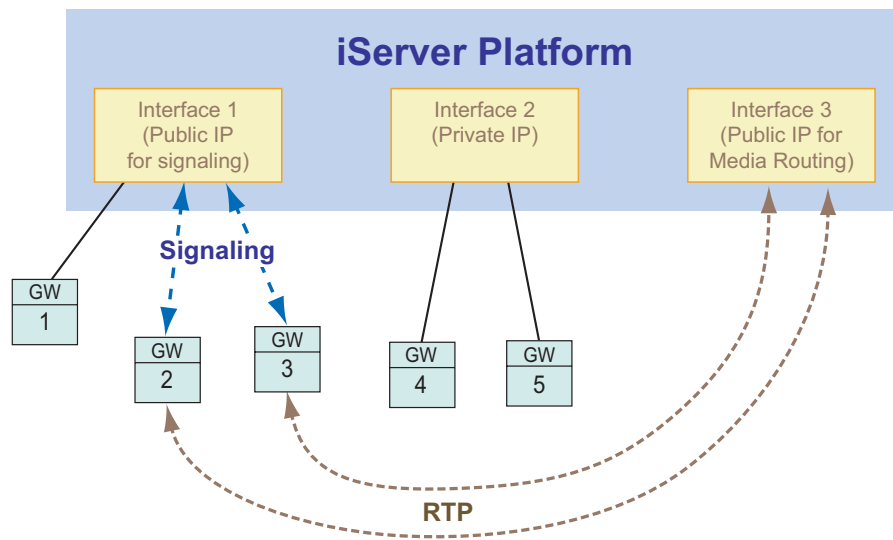
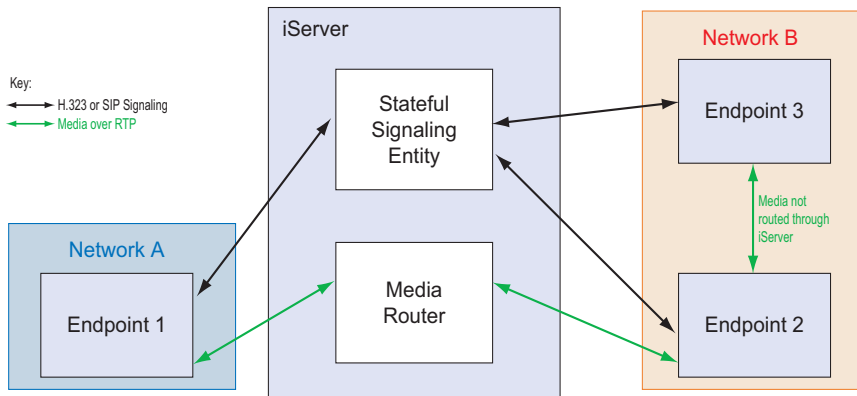




Figure 65 shows the relationship between the endpoints and MSX integrated signaling and media control components.

**Figure 65. iServer signaling and media**



The number of supported media-routed vPorts is a licensed quantity, separate from the number of *signaling* vPorts. The XML license file (`iserverlc.xml`, stored in the `servercfg` table) specifies these and other configuration parameters:

- The `MaxCalls` parameter controls the number of signaling calls licensed on an MSX. For example, `MaxCalls=100`, means that the MSX is licensed for 100 concurrent calls (media routed or signaling-only).
- The `MRVPORT` parameter controls the number of media-routed vPorts licensed on an MSX. For example, `MRVPORT=100`, for 100 media-routed calls.

### Configuring media routing

For any endpoint that is configured in the MSX database, the media routing parameter is taken from the endpoint configuration. For any endpoint that is not in the MSX database, the media routing parameter is taken from the realm configuration.

To configure NSF-based media routing, at least two physical network interfaces must be present on the MSX, along with the NSF software and feature license.

You can configure media routing to use any of the three FCE types, using the procedure in *Setting up media devices and media routing pools* on page 297 and selecting the appropriate firewall type.

## Mid-Call media change (hide address change)

During a fax call, some gateways such as Cisco 2600, may start with a certain media address/port during the initial setup, and then switch to a different port in the middle of the call. However, there are some other gateways that do not accept port switching in the middle of a call. These gateways reject such calls.

If the mid-call media change feature is enabled, for all fax calls, the MSX substitutes an assumed media address/port throughout the duration of the call, regardless of any changes that might occur with the real media address/port. Configuration of this parameter is similar to that of media routing:

- within MSX configuration
- within endpoint configuration

**WARNING:** It is generally *NOT* advisable to configure mid-call media change, unless there is a specific need (such as when a gateway is rejecting calls because the media address/port is being changed). Enabling mid-call media change increases MSX overhead, in both resources and call setup time to this endpoint, whether or not the call is a fax call.

## Preventing Rogue RTP Packets in Media Streams

RTP injection describes attempts by malicious users to inject extraneous IP packets intending to negatively impact network service quality, to deny service, and to overload network resources. The MSX provides a way to protect specifically against media vulnerabilities.

To prevent rogue RTP packets in media streams, MSX FCE software assures that only media streams with authorized packet data are accepted and routed within the MSX and media processor. The fundamental change made was to have the media processor determine the source IP from signaling information rather than from media packets as they arrive and are filtered.

### Configuring Rogue RTP Detection

You can enable and disable Rogue RTP Detection using the following global *nxconfig* command from the command line interface:

```
nxconfig.pl -e roguertpdetect -v <0/1>
```

By default Rogue RTP detection is disabled.

## Signaling Firewall Operations

To help protect your network from traffic from unauthorized sources, iServer provides two types of firewalls, one for signaling traffic and another for media traffic. This chapter describes the signaling firewall that is available on all iServers. In addition to the firewall, you can also add specific sources to a blacklist that blocks them from sending traffic to your network. Blacklisting sources is described at the end of this chapter.

The iServer's signaling firewall is automatically configured to provide firewall protection in typical iServer environments. This default configuration is derived as you install and configure your system. For many systems these default settings will be appropriate without additional configuration. However, if necessary, you can customize or extend this default firewall. The following sections describe the default components of the firewall to help you determine whether you need to make any configuration changes for your specific system.

### Firewall zones

There are three types of firewall zones, each supporting a different type of traffic: control, management, or signaling. The control firewall zone is designed for communication traffic between iServers in a redundant configurations. The management firewall zone is for communication traffic between iServer and other network entities such as RSM or an SNMP manager. The signaling firewall zone is designed for signaling traffic from SIP or H.323 endpoints. Each firewall zone is associated with a specific *service set* that defines the group of ports authorized to send traffic through it. Therefore only traffic sent to ports that are included in the appropriate service set can gain access to the network.

### Service sets

Service sets are groups of either TCP or UDP ports with an associated ICMP (Internet Control Message Protocol) rate and burst limit. The group of ports defined in a service set constitute the set of ports authorized to receive traffic on a particular firewall zone. For the ports in the service set, the ICMP rate limit defines the maximum number of ICMP Echo Requests (pings) per second

to which the iServer will respond. The burst limit lets you define an allowed increase in the rate limit, permitted on a temporary basis, to accommodate occasional fluctuations in network traffic.

iServer populates the default service set for the management firewall zone with the management interface information (management IP address) you defined during installation and the port numbers that are typically used for functions like SNMP or SSH. It also populates the default service set for the control firewall zone with the control interface information you defined if you configured your machine as part of a redundant pair. This means that when you use the default firewall entities that are provided, iServer can define your standard firewall settings without requiring you to provide additional configuration.

### ***Signaling vnets and realms***

When you create a signaling vnet you must select a signaling firewall zone to associate with it. Then when you create a realm, you must select a signaling vnet. Therefore, as you create realms, you are identifying the interfaces that should be authorized to receive traffic through your signaling firewall zone. iServer automatically adds the appropriate interfaces to the service set for the default signaling firewall zone when you create signaling vnets and realms. Again, this means that in most cases it is not necessary to configure additional firewall settings.

### ***Default configuration***

The following table summarizes the default configuration of the signaling firewall.

Firewall zone type	Default Fwzone name	Default service set	UDP ports	TCP ports	ICMP rate limits
Signaling	def_sig_zone	def_sig_set	Appropriate port numbers are added when you create a realm and specify a vnet.  For example, port 5060 for SIP traffic.	Appropriate port numbers are added when you create a realm and specify a vnet.  For example, port 5060 for SIP traffic and ports 1718, 1719, and 1719 for H.323 traffic.	rate limit: 100 burst limit: 10 (packets per second)
management	def_mgmt_zone	def_mgmt_set	161 (for SNMP)	22 (for SSH)  443 (for https-- for web connection for RSM Lite)  5432 (for RSM to communicate with the postgresSQL database for configuration and provisioning)  8081 (NexTone specific: for downloading RSM Console)	rate limit: 100 burst limit: 10 (packets per second)
control	def_control_zone	def_control_set	All ports	All ports	rate limit: 100 burst limit: 10 (packets per second)

**Note:** The default setting described in the previous sections will be appropriate for most networks. Using these default settings, the standard firewall operations are configured and enabled without requiring you to provide any additional configuration or customization. The following sections are provided if you have special requirements and should be undertaken with care to ensure that you keep firewall protection in place.

## Customizing firewall settings

The previous sections described the basic design of the signaling firewall and noted that the default configuration will be appropriate in most cases. If you have special requirements, iServer provides CLI commands to modify or extend your firewall configuration. These commands let you create additional signaling firewall zones and service sets, or modify the default ones. You cannot delete the default firewall zones or service sets because they are components of the standard firewall.

### Firewall zone command - *cli fwzone*

The firewall zone command and its options let you add new signaling firewall zones or modify existing firewall zones of any type. You can also change which service set is associated with a firewall zone. You can delete signaling firewall zones that you previously added, but you cannot delete the default firewall zones. The complete list of command options appears in the following table.

**Table 52. fwzone command**

Command	Description
<code>cli fwzone list</code>	returns a list of existing firewall zones.
<code>cli fwzone add &lt;fwzone_name&gt; &lt;type&gt;</code>	adds a new signaling firewall zone, where: fwzone_name is the name you want to assign to the new zone. type must be signaling.
<code>cli fwzone edit &lt;fwzone_name&gt; &lt;type&gt; &lt;service-set name&gt;</code>	edits an existing firewall zone, where: fwzone_name is the name of the firewall zone you want to edit. type is control, mgmt, or signaling. service-set name is the name of the service set you want to associate with this firewall zone.
<code>cli fwzone delete &lt;fwzone_name&gt; &lt;type&gt;</code>	to delete a signaling firewall zone you previously added, where: fwzone_name is the name of the firewall zone you want to delete. type must be signaling.  Note that the default control, management, and signaling firewall zones cannot be deleted.

For example, the following command adds a new signaling firewall zone named “signaling\_test” to the system:

```
cli fwzone add signaling_test signaling
```

### **Service set command - cli service-set**

The service set command and its options let you add new service sets or modify existing sets. If you add new service sets you can delete them, but you cannot delete the default signaling, management, and control service sets. The complete list of command options appears in the following table.

**Table 53. service-set command**

Command	Description
cli service-set list	returns a list of existing service sets.
cli service-set lkup <service-set_name>	lists the configured values for the specified service set, where: service-set_name is the name of an existing service set.
cli service-set add <service-set_name>	adds a new service set, where: service-set_name is the name you want to assign to the new service set.  After adding the new service set, continue defining a new service set using cli service-set edit options described in the next three rows.
cli service-set edit <service-set_name> protocol <udp/tcp> add <port>, <port>, <port-range>	edits an existing service set, where: service-set_name is the name of the service set you want to edit.  protocol is the type of transport protocol associated with the ports in the service set, either udp or tcp.  add, followed by one or more (comma-delimited) port numbers or a range of port numbers, specifies ports to include in the service set.

Command	Description
cli service-set edit <service-set_name> protocol <udp/tcp> delete <port>, <port>, <port-range>	<p>edits an existing service set, where:</p> <p>service-set_name is the name of the service set you want to edit.</p> <p>protocol is the type of transport protocol associated with the ports in the service set, either <code>udp</code> or <code>tcp</code>.</p> <p>delete, followed by one or more (comma-delimited) port numbers or a range of port numbers, specifies ports to delete from the service set.</p>
cli service-set edit <service-set_name> icmp-rate <rate, burst>	<p>edits an existing service set, where:</p> <p>service-set_name is the name of the service set you want to edit.</p> <p>icmp-rate, followed by the maximum number of pings per second to which the iServer should respond, and a burst limit to define a temporary increase in the rate limit permitted to accommodate occasional fluctuations in network traffic. The rate limit and burst limit should be separated by a comma.</p>
cli service-set delete <service-set_name>	<p>to delete a service set you previously added, where:</p> <p>service-set_name is the name of the service set you want to delete.</p> <p>Note that the default firewall zones cannot be deleted.</p>

For example, the following command adds a new service set named “test\_service\_set”:

```
cli service-set add test_service_set
```

To define that test\_service\_set include TCP ports 1 through 2500 and port 4001, but *not* include port 1500, use the following two commands:

```
cli service_set edit test_service_set protocol tcp add 1-2500, 4001
cli service_set edit test_service_set protocol tcp delete 1500
```

Finally, to define that test\_service\_set allow a maximum ping rate of 1000 per second with a burst rate of 50, use the following:

```
cli service_set edit test_service_set icmp-rate 1000,50
```



### CLI options for vnets and realms

By default, when you create a vnet, iServer associates it with the default signaling firewall zone, `def_sig_zone`. However, if you must change the firewall zone associated with a vnet, use the following CLI command:

```
cli vnet edit <vnet_name> fwzone <fwzone_name>
```

where `vnet_name` is the vnet you want to change and `fwzone_name` is the signaling firewall zone you want to associate with the vnet.

When you create a realm you select a signaling vnet to associate with it. Use the following CLI command to specify the vnet to associate with a realm:

```
cli realm edit <realmname> <vnetname>
```

where `realmname` is the name of the realm you want to adjust, and `vnetname` is the name of the vnet you want to associate with the realm.

**Note:** *Information on creating and configure signaling vnets is found in the chapter “VLAN Support”, on page 452. The complete list of signaling vnet CLI commands are listed in the section “CLI operations”.*

### Enabling or disabling the signaling firewall

By default, iServer is configured so that the signaling firewall is enabled. The option to enable/disable the firewall resides in the `server.cfg` table. If necessary you can change your configuration to disable the firewall using the following command:

```
nxconfig.pl -e ip-layer-firewall -v <value>
```

where `value` is either 1 to enable the firewall, or 0 to disable the firewall.

**Caution:** *The signaling firewall is enabled by default to ensure that firewall protection is in place. Use great care in disabling the firewall because this leaves your network open to unwanted traffic which could be harmful.*

### Blacklisting sources to prevent system access

In addition to the standard signaling firewall functions, you also have the option to block traffic from specific sources. For example, if you determine that traffic from a specific source is attempting to disrupt your network, you can blacklist that source either temporarily or permanently. These sources can be known endpoints that are already registered or configured on iServer, or unknown endpoints.

### **Blacklisting unknown sources**

iServer provides the `cli blacklist` command for you to create a list of sources you want to prevent from accessing your system. Use this command for sources that are not configured endpoints. For these unknown endpoints, you must provide the source IP address and, optionally, the source port and the destination (realm) IP address as identification of the source to blacklist. You also have the option to provide a timeout value after which time the source is removed from the blacklist. Temporary blacklisting could be useful in a case where you suspect the IP address of a source has been spoofed (forged) and is not the actual source of the unwanted traffic. In that case the unwanted traffic from that source may only be a temporary problem. The following table summarizes the blacklist command and its options.

**Table 54. blacklist command**

Command	Description
<code>cli blacklist list</code>	returns the list of blacklisted sources.
<code>cli blacklist add &lt;src-ip&gt; [src-port &lt;port&gt;] [dst-ip &lt;ip address&gt;] [timeout &lt;value in seconds&gt;]</code>	<p>adds a source to the blacklist, where:</p> <p><code>src-ip</code> is the IP address of the source.</p> <p><code>src-port</code> followed by the source port number (optional).</p> <p><code>dst-ip</code> followed by the IP address of the destination (optional).</p> <p><code>timeout</code> followed by the number of seconds that you want the source to remain on the blacklist (optional).</p> <p>You can provide <code>src-ip</code> alone, <code>src-ip</code> and <code>src-port</code>, or all three values. Providing additional values applies the blacklisting more specifically.</p>
<code>cli blacklist delete &lt;src-ip&gt; [src-port &lt;port&gt;] [dst-ip &lt;ip address&gt;]</code>	<p>removes a source from the blacklist, where:</p> <p><code>src-ip</code> is the IP address of the source.</p> <p><code>src-port</code> followed by the source port number (optional).</p> <p><code>dst-ip</code> followed by the IP address of the destination (optional).</p>

For example the following command adds to the blacklist a source (IP address 202.14.34.33, port 1719) that was sending to a specific destination (192.168.1.18). The source will remain on the blacklist for 10 minutes (600 seconds).

```
cli blacklist add 202.14.34.33 src-port 1719 dst-ip 192.168.1.18 timeout 600
```

To indefinitely blacklist all ports from the IP address 202.14.34.33 to any destination, use the following:

```
cli blacklist add 202.14.34.33
```

To blacklist port 1719 from the IP address 202.14.34.33, regardless of destination, for two hours, use the following:

```
cli blacklist add 202.14.34.33 src-port 1719 timeout 7200
```

### **Blacklisting known sources**

You have the option to blacklist endpoints that are configured on your system by enabling the blacklist property on the endpoint. Similar to unknown endpoints, you can use a timeout option to disable blacklisting of the endpoint after a specified time interval. You use options to the standard CLI command for editing endpoint configuration, `cli iedge edit`, to add or remove known endpoints from the blacklist. By default, blacklisting is disabled on an endpoint. The format of the blacklisting options for the `cli iedge edit` command follows:

```
cli iedge edit <regid> <uport> blacklist <enable/disable>
blacklisttimeout <timeout in seconds>
```

where:

`<regid>` and `<uport>` identify the endpoint you want to configure.

you can specify `enable` to blacklist the endpoint or `disable` to remove blacklisting.

`timeout in seconds` specifies the number of seconds the endpoint should remain blacklisted.

For example the following command adds an endpoint named “iedge1” to the blacklist for an hour:

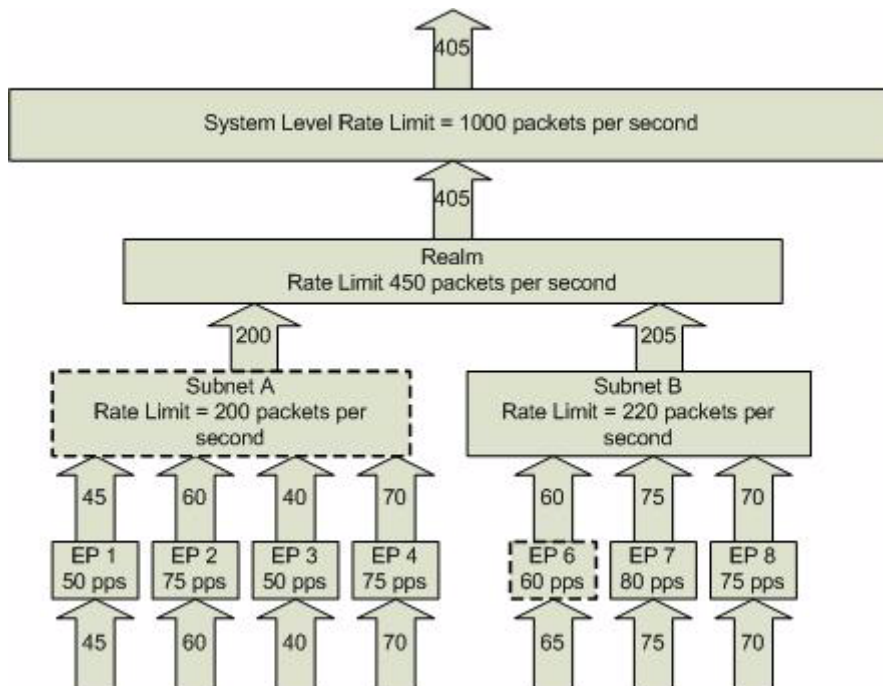
```
cli iedge edit iedge1 0 blacklist enable blacklisttimeout 3600
```

## Rate Limiting

One of the most common forms of denial of service (DoS) attack occurs when an attacker floods a network with unwanted traffic. Too much traffic at one time, even when it is caused by a burst of legitimate traffic, can impact your network operations. To help prevent problems caused by excessive traffic, MSX's rate limiting capability lets you configure acceptable limits on the rate of traffic within your network. When MSX detects traffic at rates that exceed your defined thresholds, it drops the traffic, logs information on the event and, when appropriate, generates an SNMP trap.

This highly configurable feature lets you define thresholds at both the IP and session layers and at various levels (endpoint, subnet, realm, system) within your network. The following diagram illustrates the basic concept of rate limiting. The numeric values within each box represents the input rate limit for the object. The arrows represent traffic to the object.

**Figure 66. Rate Limiting-basic input flow**



The diagram shows traffic flowing into endpoints, each with its own rate limit defined. Endpoint EP 6 is being sent traffic at 65 pps which exceeds its defined rate limit of 60 pps. Notice the traffic continuing from EP 6 to Subnet B is throttled back to the 60 pps rate limit. iServer drops the packets that exceeded the defined 60 pps threshold. Further up in the diagram, all of the endpoints within Subnet A are receiving traffic that is within their defined rate limits. However their aggregate traffic rate exceeds the limit for Subnet A. Again, the rate is throttled back to the defined limit of 200 pps and packets in excess of the defined rate limit are dropped. The traffic reaching the realm level and system levels is within the defined rate limits so iServer does not drop any packets and the rate is the same both entering and leaving the realm and system.

## IP layer rate limiting

To configure rate limiting at the IP layer, iServer provides a command to create rate limit “buckets” for different types of objects. You create buckets that contain the specific limits you want to apply. Then, using the buckets you defined, you assign the appropriate rate limit buckets to different entities in your system. The following section explains how to define rate limit buckets.

### **Defining rate limit buckets - cli ratelimitbucket**

You use the cli ratelimitbucket command to create and work with rate limit buckets. When you create buckets you specify what type of object they are for (for example, endpoint or realm) and whether the limit applies to input or output traffic. Once you create a new bucket you edit it to specify the actual limits. The complete list of command options appears in the following table.

**Table 55. ratelimitbucket command**

Command	Description
cli ratelimitbucket list	returns a list of names for existing rate limit buckets.

Command	Description
cli ratelimitbucket add <bucket_name> <mode (input/output)> <layer (ip)> <type_of_object (ep/ subnet/realm/unknown)>	<p>adds a new rate limit bucket, where:</p> <p>bucket_name is the name you want to assign to the new bucket. The name you specify must be unique.</p> <p>mode is the type of traffic to which the limit should apply, either input or output.</p> <p>layer is ip to specify the limit applies to the ip layer.</p> <p>type_of_object specifies the type of object for which this new bucket is being created, including:</p> <ul style="list-style-type: none"> <li>• ep for buckets to assign to the different types of endpoints</li> <li>• subnet for buckets to assign to subnets</li> <li>• realm for buckets to assign to realms</li> <li>• unknown for buckets to assign to endpoints that are not yet registered (unknown) on the iServer.</li> </ul> <p>To finish defining a new bucket, use the ratelimitbucket edit command options described in the four rows below.</p>
cli ratelimitbucket edit ratelimit <value>	<p>defines the limits to include in an existing rate limit bucket, where:</p> <p>ratelimit is the maximum number of packets per second you want to allow. The default is 10000.</p>
cli ratelimitbucket edit burstrate <value>	<p>burstrate defines a temporary increase in the rate limit you will permit to accommodate occasional fluctuations in network traffic. The default is 100.</p>
cli ratelimitbucket edit threshold <value>	<p>threshold is a limit, specified as a percentage beyond the defined rate limit, that defines when you want iServer to start logging information about IP packets it drops when rate limits are exceeded. The default is 0 which means that all instances are logged.</p>

Command	Description
cli ratelimitbucket edit reportingInterval <value>	reportingInterval specifies the minimum time between logging reports of traffic exceeding the rate limit. The default is 0 which means all instances are logged regardless of the frequency.  You can use threshold along with the reportingInterval to control the number of events that are logged. To limit log entries, you can use the threshold limit to create a “buffer area” above the rate limit where rate limit violations are not logged. By defining a reportingInterval you can limit the number of log entries relating to a sustained rate limit violation so that entries are only added periodically.
cli ratelimitbucket delete	to delete a rate limit bucket, where:  bucket_name is the name of the rate limit bucket you want to delete.

### **Assigning IP rate limits**

IP rate limits can be applied at various levels within your network using the rate limit buckets you created. Based on your network and the type of traffic you receive, you can select where applying rate limits will be most effective. If you want to safeguard a specific endpoint or specific subnet, you can edit its configuration to assign it a specific rate limit bucket. For realms, you can specify rate limits on both the UDP traffic and the TCP traffic it receives.

Also at the realm level, you can also assign default rate limits to use for different types of endpoints and for subnets within that realm. These rate limits are applied to endpoints or subnets that you do not configure individually. Included in the endpoint defaults is the ability to apply a rate limit to “unknown” endpoints, those that are not yet registered or configured on the iServer. For example, for unregistered endpoints you might assign a low rate limit as a precaution against unknown sources.

Finally, you can define an overall system-level rate limit that applies to the iServer as a whole. The following sections describe the commands you use to assign rate limits at different levels.

#### **Setting IP rate limits for specific endpoints**

When you configure rate limits for endpoints, you can apply rate limits on both the input traffic sent to the endpoint and the output traffic sent from the endpoint. To configure either input or output rate limits you use the cli iedge edit command to assign the endpoint an appropriate rate limit bucket.

***Setting an endpoint's input rate limit***

An endpoint's input rate limit defines the maximum number of packets per second it can receive. You assign the endpoint an input rate limit bucket to specify this limit as follows:

```
cli iedge edit <regid> <uport> max_pps_ipBucket input <bucket name>
```

where:

<regid> and <uport> identify the endpoint you want to configure.

<bucket name> specifies the name of a defined rate limit bucket containing the input limits appropriate for this endpoint.

For example, to assign an input rate limit bucket named "ep1\_input" to an endpoint named "ep1," use the following:

```
cli iedge edit ep1 0 max_pps_ipBucket input ep1_input
```

If you do not assign a rate limit bucket to an endpoint individually, its input rate limit comes from the default settings for its endpoint type, which is set in the configuration of the realm in which it resides.

***Setting an endpoint's output rate limit***

An endpoint's output rate limit defines the maximum number of packets per second it can send. You assign the endpoint an output rate limit bucket to specify this limit as follows:

```
cli iedge edit <regid> <uport> max_pps_ipBucket output <bucket name>
```

where:

<regid> and <uport> identify the endpoint you want to configure.

<bucket name> specifies the name of a defined rate limit bucket containing the output limits appropriate for this endpoint.

For example, to assign an output rate limit bucket named "ep1\_output" to an endpoint named "ep1," use the following:

```
cli iedge edit ep1 0 max_pps_ipBucket output ep1_output
```

If you do not assign a rate limit bucket to an endpoint individually, its output rate limit comes from the default settings for its endpoint type, which is set in the configuration of the realm in which it resides.

**Setting input IP rate limits for specific subnets**

When you configure rate limits for subnets, you can apply rate limits to input traffic only. The rate limit you apply defines the maximum number of packets the subnet can receive. Rate limits are not applied to subnet output traffic. To configure a subnet's input rate limit, use the `cli iedge edit` command to assign the subnet an appropriate rate limit bucket as follows:



```
cli policy edit <policy name> max_pps_ipBucket <bucket name>
```

where:

<policy name> is the name of the subnet you want to configure.

<bucket name> specifies the name of a defined rate limit bucket containing the limits appropriate for this subnet.

For example, to assign a rate limit bucket named “subnet1\_limits” to a subnet named “subnet1,” use the following:

```
cli policy edit subnet1 max_pps_ipBucket subnet1_limits
```

If you do not assign a rate limit bucket to a subnet individually, its rate limit comes from the default settings for subnets which is set in the configuration of the realm in which it resides.

### **Setting IP rate limits at the realm level**

When you configure rate limits at the realm level you can specify input rate limits that apply to the realm itself. You can define input rate limits separately for both TCP and UDP traffic. Also within your realm configuration, you can define both input and output default rate limits for different types of endpoints. For subnets, you can define a default input rate limit. iServer applies the default rate limits to endpoints and subnets that do not have their own individual rate limits configured. To configure these limits you use the `cli realm edit` command to assign an appropriate rate limit bucket to the object.

#### ***Setting a realm's input rate limits***

For realms you can define an input rate limit for UDP traffic and an input rate limit for TCP traffic. These rate limits define the maximum number of packets per second the realm can receive for each type of input. To define these limits you assign appropriate rate limit buckets to the realm. Output rate limits are not defined for realms. To assign the input limit for UDP traffic:

```
cli realm edit <realm name> max_pps_ipBucket input udp <bucket name>
```

where:

<bucket name> specifies the name of a defined rate limit bucket containing the input limits you want to apply to input UDP traffic at the realm.

To assign the input limit for TCP traffic:

```
cli realm edit <realm name> max_pps_ipBucket input tcp <bucket name>
```

where:

<bucket name> specifies the name of a defined rate limit bucket containing the input limits you want to apply to input TCP traffic at the realm.

For example, to assign a rate limit bucket called “udp\_limits” to a realm called “realm1,” that will apply to input UDP traffic on realm1, use the following:

```
cli realm edit realm1 max_pps_ipBucket input udp udp_limits
```

### ***Setting default input and output rate limits for different endpoint types***

You can assign default rate limits that apply to different types of endpoints. These default limits are applied to endpoints, according to type, when the endpoint is not specifically configured with its own rate limits. This includes defining limits for “unknown” endpoints, those that have not been registered or configured on iServer. You can assign each endpoint type both input and output rate limits in a realm’s configuration. These limits then apply to endpoints of the specified type within that realm. You can use the `cli realm edit` command to associate appropriate rate limit buckets with an endpoint type. To configure the default input rate limits for the different endpoint types:

```
cli realm edit <realm name> max_pps_ipBucket input
<phone/gateway/gatekeeper/proxy/unknown> <bucket name>
```

where:

`<realm name>` specifies the name of the realm to which you want these defaults to apply.

`<phone/gateway/gatekeeper/proxy/unknown>` are the options from which you can choose to which type of endpoint you want to assign this default rate limit.

`<bucket name>` specifies the name of the rate limit bucket that contains the rate limits you want to assign.

For example, to assign a rate limit bucket called “gateway\_limits” as the default rate limit for input on gateway endpoints on a realm called “realm1,” use the following:

```
cli realm edit realm1 max_pps_ipBucket input gateway gateway_limits
```

To configure the default output rate limits for the different endpoint types:

```
cli realm edit <realm name> max_pps_ipBucket output
<phone/gateway/gatekeeper/proxy/unknown> <bucket name>
```

where:

`<realm name>` specifies the name of the realm to which you want these defaults to apply.

`<phone/gateway/gatekeeper/proxy/unknown>` are the options from which you can choose to which type of endpoint you want to assign this default rate limit.

<bucket name> specifies the name of the rate limit bucket that contains the rate limits you want to assign.

For example, to assign a rate limit bucket called “phone\_limits” as the default rate limit for output on phone endpoints, on a realm called “realm2,” use the following:

```
cli realm edit realm2 max_pps_ipBucket output phone phone_limits
```

### ***Setting default input rate limits for subnets***

In a realm’s configuration you can assign a default input rate limit that applies to subnets within that realm. These default limits apply to subnets that are not specifically configured with their own rate limits. Output rate limits are not defined on subnets. You can use the `cli realm edit` command to associate an appropriate rate limit bucket with subnets in that realm. To configure the default input rate limits for subnets:

```
cli realm edit <realm name> max_pps_ipBucket input subnet <bucket name>
```

where:

<realm name> specifies the name of the realm to which you want these defaults to apply.

<bucket name> specifies the name of the rate limit bucket that contains the rate limits you want to assign.

For example, to assign a rate limit bucket called “subnet\_limits” as the default input rate limit for subnets on a realm called “realm1,” use the following:

```
cli realm edit realm1 max_pps_ipBucket input subnet subnet_limits
```

### **Setting IP rate limits at the system level**

To set the system level IP rate limit you use a specific, system-level option with the `cli ratelimitbucket` command. The limits you specify apply to input to the iServer as a whole and should take into account the limits your system can accommodate as well as what is appropriate from a security standpoint. Use the following to set system-level rate limits:

```
cli ratelimitbucket edit def_system_ip ratelimit <rate limit> burstrate  
<burst rate> threshold <value> reportingInterval <value>
```

where:

`ratelimit` is the maximum number of packets per second you want to allow. The default is 10000.

`burstrate` defines a temporary increase in the rate limit you will permit to accommodate occasional fluctuations in network traffic. The default is 100.

`threshold` is a limit, specified as a percentage beyond the defined rate limit, that defines when you want iServer to start logging information about IP packets it drops when rate limits are exceeded. The default is 0 which means that all instances are logged.

`reportingInterval` specifies the minimum time between logging reports of traffic exceeding the rate limit. The default is 0 which means all instances are logged, regardless of the frequency.

You can use `threshold` along with `reportingInterval` to control the number of events that are logged. To limit log entries, you can use the `threshold` limit to create a “buffer area” above the rate limit where rate limit violations are not logged. By defining a `reportingInterval`, you can limit the number of log entries relating to a sustained rate limit violation so that entries are only added periodically.

### **Enabling/disabling IP layer rate limiting**

By default, rate limiting at the IP layer is enabled. To disable or enable rate limiting at the IP layer, use:

```
nxconfig.pl -e ip-layer-rate-limiting -v <value>
```

where you use the value 1 to enable rate limiting and 0 to disable it.

### **Setting connection tracking timeout**

If traffic exceeding your configured rate limits causes UDP packets to be dropped, the connection created for the call must be closed. The connection tracking timeout interval defines how much longer a UDP connection stays open after packets are dropped. The default for this interval is 32 seconds which is appropriate for most systems. If you must change the timeout value, use the following command:

```
nxconfig.pl -e ip-layer-conntrack-timeout -v <value>
```

where `value` is the timeout interval in seconds.

### **IP rate limiting logging configuration**

You can select what type of logging you would like iServer to perform if traffic exceeding your configured rate limits causes packets to be dropped. By default, dropped packets are sent to a ULOG which is monitored by iServer. iServer’s monitoring process in turn generates SNMP traps when traffic rates warrant it. Alternatively, dropped packets can be logged in the syslog. Unless you are performing troubleshooting, the default implementation is appropriate for most networks and will generate more useful information as it ties into the

SNMP system (see the chapter “SNMP Support”, on page 116 for more information on SNMP in general). Use the following command to specify the level of logging you want to implement:

```
nxconfig.pl -e ip-layer-dropped-pkts-log -v <value>
```

where value can be one of the following:

- 0 to have no logging
- 1 to implement “LOG” logging that sends dropped packets to the syslog
- 2 to implement “ULOG” logging to send dropped packets to iServer’s monitoring process and potentially generate SNMP traps
- 3 to implement both “LOG” and “ULOG” logging

The default setting is 2.

*Note: For information on configuring SNMP-related options that are specific to rate limiting, see “Configuring SNMP-related options for rate limiting” on page 150.*

## Signaling rate limiting at the SIP session layer

*Note: Rate limiting at the signaling layer currently applies only to SIP traffic.*

In addition to IP layer rate limiting, you can implement rate limiting at the session, or signaling, layer. You can set rate limits that apply to specific SIP request methods sent to individual endpoints and realms as well as a default rate limit to apply to endpoints that you do not configure individually. You can also set an overall signaling rate limit that applies to the system as a whole. When incoming request rates exceed the limits you specified, you can configure iServer to either drop the call or send a response. If you choose to send a response, you can specify which SIP response code iServer sends.

### Setting signaling rate limits

You can define signaling rate limits that apply at different levels within your system. Based on your network and the type of traffic you receive you can choose at what level to apply rate limits and to which SIP request methods they should apply. The following sections describe the commands you use to define signaling rate limits.

#### Setting signaling rate limits for specific endpoints

When you configure signaling rate limits you specify limits that apply to specific types of SIP methods. Therefore the rate limit defines the maximum number of SIP requests of the specified type per second that the endpoint can receive. To define a signaling rate limit for a specific endpoint you must edit that endpoint’s configuration using the `cli iedge edit` command as follows:

```
cli iedge edit <regid> <uport> max_pps_sig <sip/h323> <method name>
<rate limit> <burst rate>
```

where:

<regid> and <uport> identify the endpoint you want to configure.

<sip/h323> identifies the signaling protocol (**currently SIP only**).

<method name> identifies the specific SIP method to which the limit applies:

where you can use:

inv to apply the limit to the INVITE method

reg to apply the limit to the REGISTER method

opt to apply the limit to the OPTION method

sub to apply the limit to the SUBSCRIBE method

notify to apply the limit to the NOTIFY method

info to apply the limit to the INFO method

refer to apply the limit to the REFER method

<rate limit> is the maximum number of requests per second you want to allow.

<burst rate> defines a temporary increase in the rate limit you will permit to accommodate occasional fluctuations in network traffic.

For example, to set a signaling rate limit of 1000 INVITE requests and a burst limit of 50 requests for an endpoint named “ep1,” use the following:

```
cli iedge edit ep1 0 max_pps_sig sip inv 1000 50
```

### **Setting signaling rate limits at the realm level**

At the realm level you can define a signaling rate limit that applies to the realm itself. You can also specify default rate limits for endpoints that do not have their own specific rate limits configured.

#### ***Setting a realm’s signaling rate limit***

When you configure rate limits for a realm you specify them for a specific SIP method. Therefore the rate limit defines the maximum number of SIP requests of the specified type per second that the realm can receive. To define a rate limit for a realm you must edit that realm’s configuration using the `cli realm edit` command as follows:

```
cli realm edit <realm name> max_pps_sig <sip/h323> <method name>
<rate limit> <burst rate>
```

where:

<realm name> is the name of the realm you want to configure.

<sip/h323> identifies the signaling protocol (**currently SIP only**).

<method name> identifies the specific SIP method to which the limit applies:

where you can use:

inv to apply the limit to the INVITE method

reg to apply the limit to the REGISTER method

opt to apply the limit to the OPTION method

sub to apply the limit to the SUBSCRIBE method

notify to apply the limit to the NOTIFY method

info to apply the limit to the INFO method

refer to apply the limit to the REFER method

<rate limit> is the maximum number of SIP requests per second you want to allow.

<burst rate> defines a temporary increase in the rate limit you will permit to accommodate occasional fluctuations in network traffic.

For example, to assign a signaling rate limit of 100 REGISTER requests with a burst limit of 10 requests to a realm called “realm1,” use the following:

```
cli realm edit realm1 max_pps_sig sip reg 100 10
```

### ***Setting a default signaling rate limit for endpoints***

At the realm level, you can define a default signaling rate limit for endpoints that you do not configure individually. To set a default rate limit for endpoints, you must edit the configuration of the realm that contains them using the `cli realm edit` command as follows:

```
cli realm edit <realm name> max_pps_sig_ep <sip/h323> <method name>
<rate limit> <burst rate>
```

where:

<realm name> is the name of the realm containing the endpoint for which you want to define a default signaling rate limit.

<sip/h323> identifies the signaling protocol (**currently SIP only**).

<method name> identifies the specific SIP method to which the limit applies:

where you can use:

inv to apply the limit to the INVITE method

reg to apply the limit to the REGISTER method

opt to apply the limit to the OPTION method

sub to apply the limit to the SUBSCRIBE method

notify to apply the limit to the NOTIFY method

info to apply the limit to the INFO method

refer to apply the limit to the REFER method

<rate limit> is the maximum number of requests per second you want to allow.

<burst rate> defines a temporary increase in the rate limit you will permit to accommodate occasional fluctuations in network traffic.

For example, to set a default signaling rate limit for INFO requests of 50 with a burst limit of 10 requests for endpoints within the realm “realm2,” use the following:

```
cli realm edit realm2 max_pps_sig_er sip info 50 10
```

### **Setting signaling rate limits at the system level**

To set a signaling rate limit at the system level, you use a specific, system-level option with the `cli ratelimitbucket` command that applies to the signaling (session) layer. The limits you specify apply to the iServer as a whole and should take into account the limits your system can accommodate as well as what is appropriate from a security standpoint. Use the following to set system-level signaling rate limits:

```
cli ratelimitbucket edit def_sys_session ratelimit <rate limit>
burstrate <burst rate> reportingInterval <value>
```

where:

`ratelimit` is the maximum number of requests per second you want to allow.

`burstrate` defines a temporary increase in the rate limit you will permit to accommodate occasional fluctuations in network traffic.

`reportingInterval` specifies the minimum time between logging reports of request rates exceeding the rate limit. The default is 0 which means all instances are logged, regardless of the frequency.

### **Setting reporting intervals when signaling rates exceed their limits**

For both individual endpoint and for realms, you can define how frequently iServer logs that rate limits were exceeded. You can specify a reporting interval (in seconds) that is the minimum time between reports. To specify a reporting interval for an individual endpoint, edit its configuration as follows:

```
cli iedge edit <regid> <uport> reportingInterval <number of seconds>
```

where:

<regid> and <uport> identify the endpoint you want to configure.

`reportingInterval <number of seconds>` specifies the minimum time, in seconds, between logging that request rates exceeded the rate limit. The default is 0 which means all instances are logged, regardless of the frequency.



To specify a reporting interval for a realm, edit its configuration as follows:

```
cli realm edit <realm name> reportingInterval <number of seconds>
```

where:

`<realm name>` identifies the realm you want to configure.

`reportingInterval <number of seconds>` specifies the minimum time, in seconds, between logging that request rates exceeded the rate limit. The default is 0 which means all instances are logged, regardless of the frequency.

For example, to define a reporting interval of 60 seconds for a realm named “realm1,” use the following:

```
cli realm edit realm1 reportingInterval 60
```

**Note:** *Reporting interval can also be set at the system level and is defined when you configure system level rate limits. See “Setting signaling rate limits at the system level” on page 360 for more information.*

### **Setting the drop policy for rate limiting at the SIP session layer**

If the incoming signaling rate exceeds the limits you define, you can choose to have iServer either drop the calls or respond with a message. If you choose to respond you can configure which SIP message iServer sends. These options are set at the realm level. Use the following commands to set your session rate limiting policy:

```
cli realm edit <realm name> sessionLayerRateLimitingPolicy <value>
```

where:

`<realm name>` is the name of the realm you want to configure.

`sessionLayerRateLimitingPolicy <value>` can be either 0 to drop the calls, or 1 to respond to the calls. The default value is 0.

If you choose to respond when rate limits are exceeded, use the following command to specify which SIP response code you want iServer to send:

```
cli realm edit <realm name> sessionLayerRateLimitingErrorResponse  
<value>
```

where:

`<realm name>` is the name of the realm you want to configure.

`sessionLayerRateLimitingErrorResponse <value>` is the SIP response you want to send. The default is 503 (Service Unavailable). You can specify any defined SIP response code number.

***Enabling/disabling session layer rate limiting***

By default, rate limiting at the session layer is disabled (0). To disable or enable rate limiting at the session layer, use:

```
nxconfig.pl -e session-layer-rate-limiting -v <value>
```

where you use the value 1 to enable rate limiting and 0 to disable it.

# **Part 3:**

## **RADIUS AAA, 3GPP Rx, and Billing**

## RADIUS AAA Services

---

### Introduction

Radius is a client/server protocol for providing security-related services Authentication, Authorization, and Accounting (AAA) and Packet of Disconnect (POD). The MSX acts like a Radius client.

### Supported services

#### **Authentication**

Support for RADIUS-based IP, ANI, and digest authentication are available for SIP calls; IP and ANI, for H.323 calls. Cisco Access Token (CAT) via Challenge Handshake Authentication Protocol (CHAP) is supported only for SIP calls. Both call authentication services allow for storing user authentication data in a central location, separate from the MSX. See *Authentication details* on page 369 for more information.

#### **Authorization**

This service allows the MSX along with the Radius server to authorize calls and limit call duration, based on prepaid service remaining balances. Authorization applies to both SIP and H.323 users.

#### **Accounting**

Support for sending RADIUS accounting messages is also configurable for the MSX. This function transmits call-related data to a RADIUS server.

See *Accounting details* on page 379 for more information.

Through POD, the RADIUS server transmits a specialized packet to the MSX RADIUS client that forces call disconnection. This is useful in scenarios like prepaid wholesale service, where a customer/retailer has pre-purchased an amount of service (typically in currency or minutes), or in scenarios that must limit the cumulative total call duration allowed from a particular gateway.

When the MSX receives the POD from the RADIUS server, it validates the packet, then disconnects the call.

## Global settings

Global parameters related to Radius configuration are set by running the `nxconfig.pl` utility. To run `nxconfig.pl`:

1. Log onto the MSX as root.
2. Enter the `nxconfig.pl` commands indicated in the following sections:

### ***Enabling RADIUS authentication***

RADIUS authentication must be enabled for allow processing of SIP and H.323 calls for billing. By default, authentication is disabled for both. To enable RADIUS authentication for SIP and H.323 calls set the `billingtype servercfg` attribute to `ciscoprepaid`. This attribute also enables support for prepaid services.

To enable support for SIP and H.323 services type the following command:

```
nxconfig.pl -e billingtype -v ciscoprepaid
```

See *Authorization details* on page 376 for a description of the prepaid service support feature.

### **Setting the authentication username & password by CLI**

Set the account name and password on the RADIUS server that the MSX will operate under during authentication operations for prepaid service using the following commands:

```
nxconfig.pl -e first-authpass -v userid
nxconfig.pl -e first-authpass -v password
```

where userid is the authentication user ID for the MSX and password is the password for userid.

### **Setting the authentication username & password by RSM Console**

To configure these credentials using RSM Console, see *Global configuration using RSM Console* on page 54. Look for Table 4: Billing tab, on page 54.

**Setting the authorization username & password by CLI**

The account name and password on the RADIUS that the MSX will send during authorization operations for prepaid service. To set these attributes, enter:

```
nxconfig.pl -e second-authpass -v password
nxconfig.pl -e second-authpass -v userid
```

where userid is the authorization user ID for the MSX and password is the password for userid.

**Setting the authorization username & password by RSM Console**

To configure these credentials using RSM Console, see *Global configuration using RSM Console* on page 54. Look for Table 4 Billing tab, on page 54.

**RADIUS accounting**

The `radacct servercfg` attribute enables RADIUS accounting message logging. To enable this, enter:

```
nxconfig.pl -e radacct -v 1
```

Restart MSX when prompted.

**Note:** *In-process H.323 calls, all incomplete call setups, are dropped during a restart.*

**RADIUS database directory**

This parameter sets the disk location for RADIUS processes to store data temporarily while in process. You should set this to an absolute path to your chosen storage directory, in a disk partition which has a substantial amount of available space. The size of the RADIUS database will grow and shrink dynamically depending on system load, whether receiving processes are running, etc., so be sure to provide at least a couple of Gigabytes of storage for it. To set this directory, enter:

```
nxconfig.pl -e raddir -v path
```

where path is the path to the RADIUS log directory. Restart MSX when prompted.

**Note:** *In-process H.323 calls, all incomplete call setups, are dropped during a restart.*

**Configuring RADIUS-MSX communication**

Communication between the MSX and the RADIUS server is set up as noted in this section.

Be sure to restart MSX when prompted after making any of the following RADIUS-related changes.

*Note: In-process H.323 calls, all incomplete call setups, are dropped during a restart.*

**Example 1, POD disabled:** Here is a sample of the `nxconfig.pl` commands to enter to set up RADIUS support with the POD feature disabled:

```
nxconfig.pl -e first-radserver -v primary rad server IP
nxconfig.pl -e first-radsecret -v primary shared secret
nxconfig.pl -e second-radserver -v secondary rad server IP
nxconfig.pl -e second-radsecret -v secondary shared secret
nxconfig.pl -e radtimeout -v timeout seconds
nxconfig.pl -e radretries -v retry limits
nxconfig.pl -e raddeadtime -v dead-time seconds
nxconfig.pl -e radacct -v 1
nxconfig.pl -e raddir -v path
nxconfig.pl -e podsupport 0
```

**Example 2, POD enabled:** Here is a sample of the `nxconfig.pl` commands to enter to set up RADIUS support with the POD feature enabled and configured:

```
nxconfig.pl -e first-radserver -v primary rad server IP
nxconfig.pl -e first-radsecret -v primary shared secret
nxconfig.pl -e second-radserver -v secondary rad server IP
nxconfig.pl -e second-radsecret -v secondary shared secret
nxconfig.pl -e radtimeout -v timeout seconds
nxconfig.pl -e radretries -v retry limits
nxconfig.pl -e raddeadtime -v dead-time seconds
nxconfig.pl -e radacct -v 1
nxconfig.pl -e raddir -v path
nxconfig.pl -e podsupport 1
nxconfig.pl -e podport -v port
nxconfig.pl -e podusername -v userid
nxconfig.pl -e podserverkey -v server key
```

As the examples show, one RADIUS server (or optionally two) can be specified, along with the shared authentication/encryption key (the `radsecret` attribute) for each server. The other attributes apply to both named servers.

The MSX sends AAA requests to the first RADIUS server (`first-radserver`) and waits (`radtimeout seconds`) for a response. If it fails to get a response, it retries (`radretries times`). It then attempts to send the request to the second RADIUS server, `second-radserver` (if one is configured).

If the MSX continues to get no response from a RADIUS server for `raddeadtime` seconds, it marks the server as failed and does not send any requests to that server for another `raddeadtime` seconds.

Note that in the event any record cannot be transmitted to any of the listed servers, it is retained until communication between the RADIUS client and a server is restored, and the record is successfully sent. In this way, no RADIUS record is ever lost. See *RADIUS database directory* on page 366 for considerations concerning this.

Table 56 lists all the configurable RADIUS parameters and their definitions.



**Table 56. RADIUS Parameters**

Parameter Name	Description	Default Value
first-radserver second-radserver	The IP addresses of the primary and secondary RADIUS servers to which the client will send records. Only the primary server is used for SIP authentication.	n/a
first-radsecret second-radsecret	The shared authentication and encryption keys for all communications between the RADIUS client and the primary and secondary RADIUS servers. The RADIUS server key has no default value; however, the key must match the encryption key used on the RADIUS server.	n/a
radtimeout	The interval (in seconds) a the MSX waits for a server host to reply. Applies to all RADIUS servers.	5
radretries	The number of times a RADIUS request is resent to a server before concluding the server is not available. (This is usually invoked when a server is not responding, or responding too slowly as defined by <i>radtimeout</i> , above.)	3
raddeadtime	The duration (in seconds) that a server is considered unavailable when it times out (based on <i>radtimeout</i> and <i>radtries</i> ) before attempting to reach it again.	
radacct	A switch to turn the MSX RADIUS accounting “on” or “off”.	
raddir	The path to the directory into which RADIUS log files are placed.	
podsupport	Enables or disables POD support for the cluster.	
podport	Specifies the port number to which the RADIUS server must send its POD packets.	1700
podusername	The userid the RADIUS server sends to the iServer when sending POD packets.	
podserverkey	The shared secret text string between the iServer and the RADIUS server.	

## Authentication details

**Note:** *MSX supports centralized call authentication via RADIUS only, not local system access authentication.*

SIP provides that certain network entities can require authentication from SIP UAs registering with, or inviting, calls through them. The MSX can be configured to perform digest<sup>1</sup> authentication to establish the user's right (at the endpoint/uport level) to use the MSX.

Sets of username-password combinations, stored in a centralized remote RADIUS database to which the MSX has access, are used to establish the endpoint's right to use the services of the MSX.

*Note: For clarity, in the example flows given here, messages such as ACK, Trying, Ringing, OK, etc.) are not shown.*

*Once the INVITE has reached the endpoint, call flows proceed as they would on systems without SIP Authentication, so the diagrams stop once the INVITE has reached the egress point/called endpoint.*

### **SIP requests challenged**

Some messages make sense to challenge; others (such as ACK) do not. The SIP requests to be challenged are configurable on a per realm basis.

Challenges supported include those listed in Table 57 on Page 375, provided:

- SIP authentication is enabled, and
- the given realm is configured to challenge them (see *Configuring Challenging (SIP), by Realm* on page 374).

Note that ACK and CANCEL messages are never challenged.

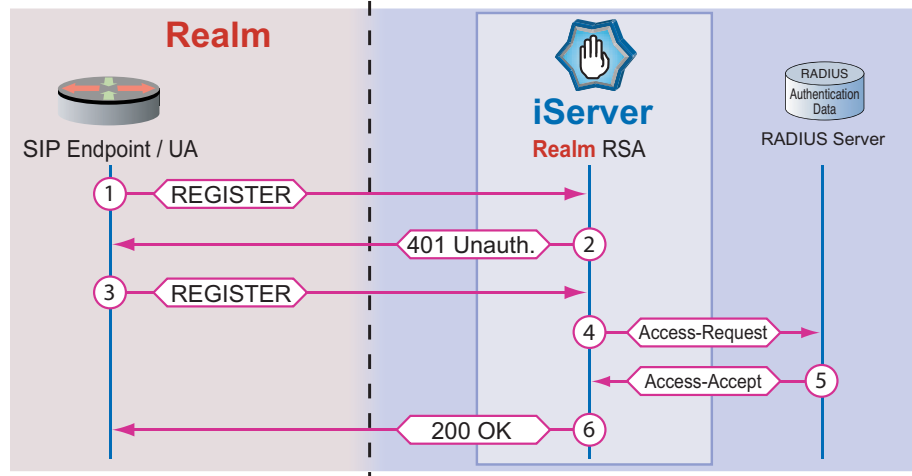
### **Registration flow**

The flow for endpoint registration under this feature involves extra messages and headers (compared to the flow without authentication), as noted in the flow below. A description of the steps leading to a successful registration follows. The figure below illustrates this flow, which is basically REGISTER, 401 Unautho-

1. In "digest" authentication, the username and password tokens are encrypted, and a "nonce" value (i.e., one intended for one-time use) is provided by the server, to prevent re-use of even the encrypted tokens.

rized, REGISTER (again, with credentials), 200 OK (success). Figure 67 illustrates this flow.

**Figure 67. SIP Authentication Registration Flow**



### **Details**

The SIP UA sends its REGISTER request to the MSX, which the MSX challenges by responding with a 401 Unauthorized, containing the nonce to be used by the UA when registering. The UA sends a second REGISTER, but this time includes the authentication response based on the nonce it received in the MSX 401. The MSX decodes the response, and interacts with the RADIUS server, using Access-Request and Access-Accept messages to validate the credentials. This time the MSX responds to the UA with 200 OK, indicating the registration succeeded.

Note that the second REGISTER (the one sent in response to the 401) contains additional information along with the challenge response, including the nonce it used, and the realm the endpoint is configured into (as shown in the samples below.)

### **Sample registration headers**

The 401 Unauthorized response returned by the MSX if SIP Authentication is enable contains a WWW-Authenticate header. For example:

```
WWW-Authenticate: Digest realm="NexTone",
nonce="61840ad2070", stale=FALSE, algorithm=MD5
```

The endpoint's second REGISTER contains an additional Authorization header. For example:

```
Authorization: Digest username="foo", realm="NexTone",
nonce="61840ad2070", response="edd11dd6f40aa7db890e3c86b7b43
9c5", uri="sip:204.180.228.157", algorithm=MD5
```

### **Registration failure**

A registration fails if authentication fails (for invalid user or bad password), in which case the MSX responds with another 401 Unauthorized.

### **Call flows**

This section discusses call flow scenarios covered by RADIUS authentication services. Each subsection reflects a different type of network architecture.

The flows for call set-up with RADIUS Authentication enabled are somewhat different than a call set-up without it. The flow includes additional messages and headers, which this section highlights. The precise differences depend on the network devices' configuration.

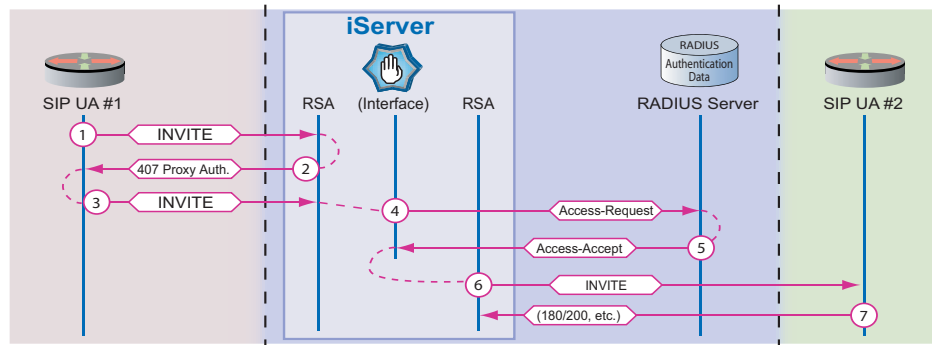
### **Basic call flow with RADIUS authentication**

*Note: RADIUS remote is currently the only supported validation method. Local validation is not yet supported.*

In this flow, instead of validating the user-pw data for an endpoint using data stored in its own database, the MSX passes the received tokens to a RADIUS server. The RADIUS server performs the validation, and returns either an Access-Accept, or Access-Reject.

The basic call set up flow (similar to the registration flow shown above, but with remote validation), is illustrated in Figure 67.

**Figure 68. Call Flow with RADIUS Authentication**



**Authentication failure**

If the RADIUS server rejects the credentials forwarded from the UA second INVITE, the response in message 5 is instead *Access-Reject*, and the call setup fails.

**Proxy-Based Call Flows**

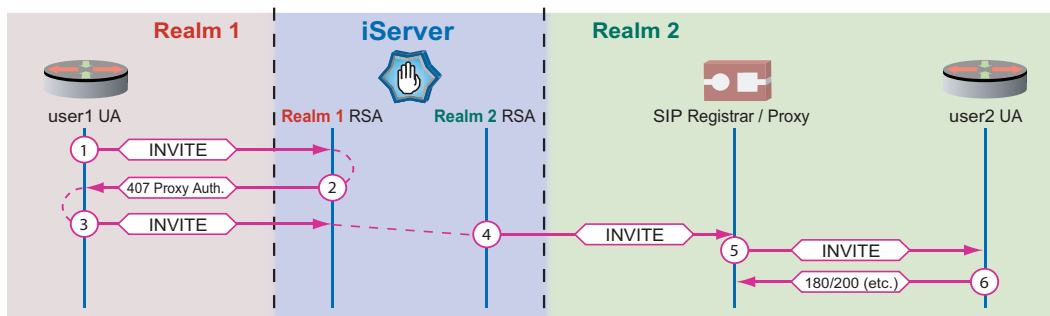
The MSX, if so configured, requires authentication. Far-end proxies can require it also, which requires a different flow from far-end proxies that do not. These two scenarios are discussed below, followed by the possible failure scenarios and how they are handled.

**Call with one proxy authentication**

In this scenario, the MSX has RADIUS Authentication enabled, but the far-end proxy does not.

This flow is similar to the registration flow given above; i.e., INVITE, 401, INVITE (again), INVITE (forwarded). Figure 68 illustrates this. Two additional messages (2 and 3) are required to complete this scenario compared with the flow of a call without authentication. See Figure 69.

**Figure 69. SIP Authentication for One Proxy**



Note that this is the flow for the *MSX* performing the authentication; if the far-end proxy had been the one performing the authentication, the two additional messages (2 and 3) would have passed between the *MSX* and the far-end registrar, instead of the originating UA and the *MSX*.

The INVITE initially sent by an endpoint normally does not contain an authorization header<sup>2</sup>. The *MSX* responds with *407 Proxy Authentication Required*,

2. In some exceptional cases, such as a hacking attempt or software malfunction, the initial INVITE actually may contain credentials, but by definition, they won't have a correct nonce, so authentication will fail.

containing a Proxy-Authenticate header, with the nonce to be used in the endpoint's repeat INVITE. For example:

```
Proxy-Authenticate: Digest realm="NexTone",
nonce="61840aa6287", stale=FALSE, algorithm=md5
```

The endpoint repeats its INVITE, this time including the Proxy-Authorization header based on the received nonce. The response parameter contains the calculated authentication data for the MSX. For example:

```
Proxy-Authorization: Digest username="foo", realm="NexTone",
nonce="61840aa6287", response="4c5a4e3d300eca2b580af95a65c01
b11", uri="sip:6135551234@204.180.228.157", algorithm=md5
```

This second INVITE meets the authentication requirements, and the call setup proceeds. The remainder of the setup is the same as for SIP calls made on systems without SIP Authentication enabled.

### **Configuring SIP authentication**

This section details the procedure of configuring an MSX for RADIUS-based SIP Authentication. Two areas need configuration: the server, and the user agents (UAs).

#### **Enabling MSX challenging**

This feature is enabled or disabled from RSM Console **iServerConfiguration** panel, the **SIP** tab, SIP Authorization pull-down. For this release, the only options are RADIUS and none.

#### **Configuring Challenging (SIP), by Realm**

Challenging is enabled by realm and by message type, using CLI or RSM Console:

##### ***CLI syntax***

The CLI command for SIP authentication has the following syntax:

```
cli realm edit realm name sipauth [methods]
```

##### ***CLI details***

The supplied parameters of the command break out as shown in Table 57.

**Table 57. sipauth CLI Command Syntax**

Parameter	Description	Options
<u>realm name</u>	The literal name for the realm, assigned when it was created.	—
<u>methods</u>	The method(s) to challenge	<ul style="list-style-type: none"> <li>• <code>inv</code> – INVITE</li> <li>• <code>reg</code> – REGISTER</li> <li>• <code>bye</code> – BYE</li> <li>• <code>info</code> – INFO</li> <li>• <code>opt</code> – OPTIONS</li> <li>• <code>sub</code> – SUBSCRIBE</li> <li>• <code>notify</code> – NOTIFY</li> <li>• <code>refer</code> – REFER</li> <li>• <code>all</code> – all message types supported by this feature (i.e., the ones listed above)</li> <li>• <code>none</code> – disable challenging</li> </ul>

***RSM Console procedure***

The SIP methods to challenge are set in the **Add Realm** and **Modify Realm** windows. (Open RSM Console, double-click the MSX icon, from the **iServer** window, select Utilities > Realm, then either double-click the existing realm, or choose Edit > Add.)

From the **Add/Modify Realm** window, locate the SIP Authentication list, and select the method(s) to subject to RADIUS authentication. To select more than one method, hold down the <Ctrl> key while clicking on each item, or select all.

**Configuring user agents**

Each SIP user agent (UA)/endpoint must have an account on the RADIUS authentication server. Adding the account to the RADIUS server must be done according to the RADIUS server procedure.

**Authentication From-header cross check**

The MSX can be configured to cross-check the user name used to authenticate a call (`Authentication Name`) against the RADIUS user name (the part preceding the @ sign, usually a phone number) in a call's `From` header. If the two do not match, the call is rejected.

To enable this feature, set the `matchsipauthuser` parameter with `nxconfig.pl`, by entering:

```
nxconfig.pl -e matchsipauthuser -v 1
```

Disable the feature by entering:

```
nxconfig.pl -e matchsipauthuser -v 0
```

### **Authentication caveats**

When setting up SIP Authentication, take note of the items listed in this section.

- The primary server and secret (password) On the Billing tab of RSM Console **iServerConfiguration** window must be configured for this feature to work.
- While the RADIUS package the MSX uses includes a server, *this server should not be run locally for performing SIP authentication*. Doing so will adversely impact iServer call-carrying performance and capacity.
- The RADIUS server must support the IETF document, `draft-sterman-aaa-sip-00.txt`.
- The RADIUS server must be configured to accept and process digest authentication requests from the MSX. The procedure to configure this depends on the RADIUS server being used.

### **Authorization details**

If **authentication** says that a user may *use* the network controlled by the MSX, **authorization** says what he may *do* with it.

The MSX SIP Authorization feature provides support for *prepaid* service; that is, a customer purchases a block of transport time against which calls carried are subsequently debited. With the prepaid feature enabled, the MSX meters the duration of an active session after an end user is authenticated and authorized. The call is released if the account limit is reached while the call is still in progress.

Authorization for prepaid service happens in two phases:

1. **Calling Party.** When an end user call setup request is received, the MSX sends calling party info and a password to the RADIUS server, which validates the user credentials. If authentication succeeds, the RADIUS server returns the amount of credit (in currency; e.g., dollars) available to that calling customer. If the RADIUS server responds with an `Access-`

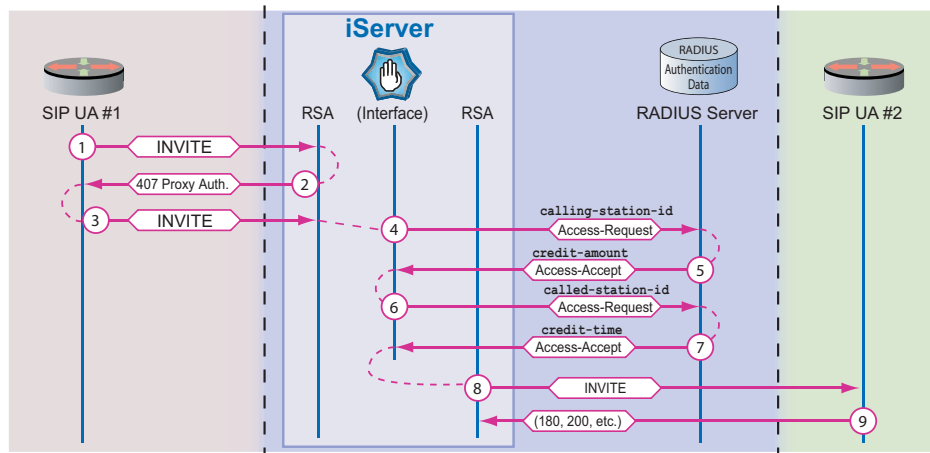


Reject, or with an Access-Accept that either does not contain an h323-credit-amount attribute, or one that indicates a credit amount of zero, the call is rejected.

2. **Called Party.** If the calling user passes phase 1 authorization, the MSX sends another Access-Request with *called* party information to the RADIUS server. Based on the calling and called party information, the RADIUS server returns the credit time (in seconds) remaining on that customer's account. If the RADIUS server responds with an Access-Reject, or with an Access-Accept that either doesn't contain an h323-credit-time attribute, or contains one indicating a time of zero, the MSX rejects the call.

Figure 70 shows the flow associated with this feature. Transactions 4 and 5 cover phase 1; 6 and 7, phase 2.

**Figure 70. RADIUS Authorization for Prepaid Service**



### Configuring prepaid services

See *Enabling RADIUS authentication* on page 365 for setting up prepaid service support.

### POD details

POD is enabled at the MSX level, through the `billingtype servercfg` attribute mentioned above, and the `podsupport` attribute detailed below.

The actual “Packet of Disconnect” is a RADIUS `access-request` packet sent from a RADIUS server used to release active calls. In release 4.1, the Cisco POD approach is supported. The POD packet includes the following two vendor-specific attributes (VSAs):

- `h323-conf-id`, with the same content as received from the gateway for this call when it was set up.
- `h323-call-origin`, with the same content as received from the gateway for the call leg of interest.

Once the POD is validated, the MSX sends an `access-accept` to the RADIUS server, and releases the call. If there are multiple calls for a customer whose account runs dry (for example), all the calls for that vendor are dropped, when the RADIUS server sends a POD for each call carried for that customer.

### **POD settings**

The global settings described above (*Global settings* on page 365) include four settings to configure POD support on the MSX. To run these settings, log into the MSX as root and enter the command shown.

*Note: Be sure to restart MSX when prompted after changing POD settings. In-process H.323 calls, all incomplete call setups, are dropped during a restart.*

- **POD support** – Set to `enable` (1) to turn on POD support.

```
nxconfig.pl -e podsupport 1
```

- **POD port** – The MSX interface port number on which Packets of Disconnect are accepted.

```
nxconfig.pl -e podport -v port
```

- **POD username** – The MSX account name that is authorized to execute POD functionality. Note that spaces are allowed, because the parameter value is quoted.

```
nxconfig.pl -e podusername -v userid
```

- **POD server-key** – the shared secret, between the RADIUS server and MSX, for disconnecting calls.

```
nxconfig.pl -e podserverkey -v server key
```

### **POD caveats**

- For POD services, RFC 3576 is not supported in 4.3.

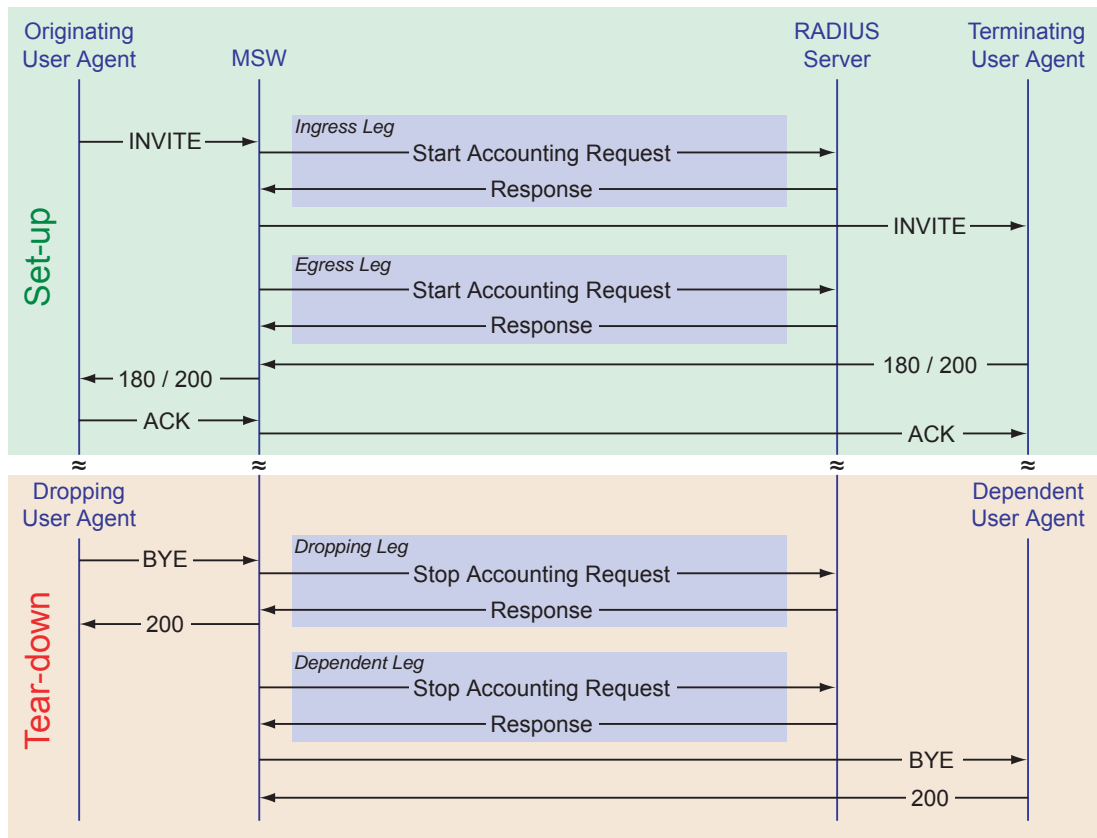
## Accounting details

Once **authentication** and **authorization** have taken place, **accounting** records are generated to record how the allocated resources were actually used. On the MSX, a subset of call parameters can be sent to external servers via its RADIUS client, using the RADIUS data transmission protocol. The server(s) to which the records are sent are specified in *Configuring RADIUS-MSX communication* on page 366.

### RADIUS event sequence

Figure 71 depicts the sequence of events resulting in accounting data being collected on the RADIUS server(s). Although this generalized example uses the SIP model, this sequence applies to H.323 as well. The data is transmitted to the RADIUS server upon call termination.

**Figure 71. RADIUS Accounting, Event Sequence**



**RADIUS record layouts**

The RADIUS records containing CDR data come in two different record layouts described below:

- Cisco XA3+ <sup>3</sup>
- Cisco IOS 12.2(11)T

These record formats are documented in the two sub-sections following “Setting the record type”. The tables in the subsections cross-reference the corresponding CDR fields, where applicable.

**Setting the record type*****nxconfig.pl procedure***

To specify your record format, log into the MSX and enter:

```
nxconfig.pl -e acctsessionid-overloaded -v 1
```

Restart MSX when prompted.

**Note:** *In-process H.323 calls, all incomplete call setups, are dropped during a restart.*

***RSM Console procedure***

To select the record type using RSM Console, go to the **iServerConfiguration** window Billing tab, and locate the Use Overloaded Session ID Format checkbox. Place a check in the box for XA3+ format, or clear the check for 12.2(11)T format.

**Radius XA3+ record layout**

Table 58 details the fields in the XA3+ record layout.

---

3. RSM Console uses the term, *Overloaded Session ID Format* for XA3+ format.

**Table 58. RADIUS XA3+ Record Fields**

Field No.	Name	Description	CDR Field No.
1	Called-Station-Id	The number to which the call was placed.	9
2	Calling-Station-Id	The number from which the call was placed	18
3	Acct-Status-Type	Indicates whether this record marks the beginning of user service ( <i>Start</i> ) or the end ( <i>Stop</i> ).	n/a
4	Acct-Delay-Time	The number of seconds the client has been trying to send this record.	
5	Acct-Authentic	The source of authority from which the user was authenticated.  Valid values: <i>Local</i> (MSX), <i>RADIUS</i> (a RADIUS server), or <i>Remote</i> (another remote authentication server type).	
6	Acct-Session-Id	A unique Accounting ID used to match start and stop records in a log file. The start and stop records for a given session have the same Acct-Session-Id. See Table 59 for how this field is constructed.	n/a
7	Acct-Session-Time	<i>Note: This field is only present in records where Acct-Status-Type equals Stop.</i>  Indicates the number of seconds the user received service.	n/a

Field No.	Name	Description	CDR Field No.
8	Service-Type	<p>Type of service used in this call.</p> <ul style="list-style-type: none"> <li>In a request, the service requested, as follows:               <p>Login-User: the user requests host access on the NAS.</p> <p>Framed-User, requests a PPP or SLIP (framed) protocol connection.</p> <p>Administrative-User, requests permission to execute privileged commands.</p> </li> <li>In a response, the service provided:               <p>Login-User: the user was connected to a host.</p> <p>Framed-User: Framed service (SLIP or PPP) is being provided.</p> <p>Administrative-User: privileged access is available.</p> <p>Exec-User, EXEC (user shell) access is granted on the NAS.</p> </li> </ul>	
9	NAS-IP-Address	The IP address of the NAS used for this call.	
10	NAS-Port	Specifies the slot the call came in on.	
11	NAS-Port-Type	The type of the physical port of the NAS that is authenticating the user. Valid values include Sync, Async, ISDN, ISDN-V110, and ISDN-V120.	
12	NAS-Identifier	A string identifying the NAS originating the access request.	
13	Proxy-State	Currently, this field is always blank.	
14	Connect-Info	Provides information about connection speeds, modulation, and compression for modem dial-in connections. Currently, this field is always blank.	

The field separator is a comma character: “,”. See Table 15 on page 396 for details on CDR fields.

Note that the *Acct-Session-Id* field is a compound field, containing a concatenation of several other fields (such as call start and end times and dates), with the virgule (“slash” character, /) as the field separator. As noted above, it is simply a unique field used for call start and stop record matching.

***Radius XA3+ Acct-Session-Id field layout***

Table 59 details the sub-fields that make up the *Acct-Session-Id* field in the XA3+ record layout. The field separator between these sub-fields is the virgule (“slash”) character.

**Table 59. RADIUS XA3+ Acct-Session-Id Sub-Fields**

Field No.	Description
1	An incrementing hexadecimal number
2	Time, Zone, and Date (separated by spaces)
3	Call source reg_id
4	Conference ID
5	answer OR originate
6	VoIP OR Telephony
7	(not used)
8	(not used)
9	(not used)
10	Call source IP address
11	Conference ID

***Sample XA3+ records***

Below are two sample CDRs (one Start, one Stop), with the *Acct-Session-Id* field marked in underlined blue:

```
6001,7812345678,Start,0,Local,37f/11:19:43.853 EDT Tue Aug 09
2005/csPhone/000653dc-3ede0743-0e21596f-
3acd108c@172.16.1.171/answer/VoIP////172.16.1.171/000653dc-3ede0743-
0e21596f-3acd108c@172.16.1.171,Login-
User,212.187.230.148,0,Async,msw,0x,
```

6001,7812345678,Stop,0,Local,[37f/11:19:43.853 EDT Tue Aug 09 2005/csPhone/000653dc-3ede0743-0e21596f-3acd108c@172.16.1.171/answer/VoIP/11:20:02.671 EDT Tue Aug 09 2005/11:20:02.671 EDT Tue Aug 09 2005/7f/172.16.1.171/000653dc-3ede0743-0e21596f-3acd108c@172.16.1.171](#),0,Login-User,212.187.230.148,0,Async,msw,0x,

### **Radius 12.2(11)T layout**

Table 12 details the fields in the *12.2(11)T* record layout. Note that there are two different record layouts, based on the two accounting status types (Acct-Status-Type of either Start or Stop). The Stop record has two additional fields, Acct-Session-Time (shown as field no. in Table 12), and Vendor-Specific (shown as field no. in Table 12), which the Start record does not have.

Also note that the layout includes a block of attribute-value pairs following the Vendor-Specific field (field ), each of which consists of a field name, an equal sign, and an optional value for the named parameter. These are *vendor-specific attributes* (or VSAs) provided for in RFC 2865 and 2866. For parameters with no value in a particular record, the equal sign is followed directly by the field-separator character, the comma. Some of these pairs are specific to Cisco, the rest are specific to NexTone, as noted in the table.



**Table 12. RADIUS 12.2(11)T Record Fields**

Field No.	Name	Description
1	Called-Station-Id	The number to which the call was placed. Taken from CDR field number 9.
2	Calling-Station-Id	The number from which the call was placed
3	Acct-Status-Type	Indicates whether this record marks the beginning of the user service (Start) or the end (Stop).
4	Acct-Delay-Time	The number of seconds the client has been trying to send this record.
5	Acct-Authentic	The source of authority from which the user was authenticated.  Valid values: <i>Local</i> (the MSX), <i>RADIUS</i> (a RADIUS server), or <i>Remote</i> (another remote authentication server type).
6	Acct-Session-Id	A unique Accounting ID used to match start and stop records in a log file. The start and stop records for a given session have the same Acct-Session-Id.
7	Acct-Session-Time	<i>Note: This field is only present in records where Acct-Status-Type equals Stop.</i>  Indicates the number of seconds the user has received service.

Field No.	Name	Description
8	Service-Type	<p>Type of service used in this call.</p> <ul style="list-style-type: none"> <li>In a request, the service requested, as follows: <ul style="list-style-type: none"> <li>Login-User: the user requests host access on the NAS.</li> <li>Framed-User, requests a PPP or SLIP (framed) protocol connection.</li> <li>Administrative-User, requests permission to execute privileged commands.</li> </ul> </li> <li>In a response, the service provided: <ul style="list-style-type: none"> <li>Login-User: the user was connected to a host.</li> <li>Framed-User: Framed service (SLIP or PPP) is being provided.</li> <li>Administrative-User: privileged access is available.</li> <li>Exec-User, EXEC (user shell) access is granted on the NAS.</li> </ul> </li> </ul>
9	NAS-IP-Address	The IP address of the NAS used for this call.
10	NAS-Port	Specifies the slot the call came in on.
11	NAS-Port-Type	The type of the physical port of the NAS that is authenticating the user. Valid values include Sync, Async, ISDN, ISDN-V110, and ISDN-V120.
12	NAS-Identifier	A string identifying the NAS originating the access request.
13	Proxy-State	Currently, this field is always blank.
14	Connect-Info	Provides information about connection speeds, modulation, and compression for modem dial-in connections. Currently, this field is always blank.
<b>Cisco VSAs</b>		
15	Vendor-Specific	<p><i>Note: This field is only present in records where Acct-Status-Type equals Stop.</i></p> <p>Indicates the start of a block of vendor-specific fields. Its length is 8 bytes, and its value can be ignored.</p>

Field No.	Name	Description
16	h323-incoming-conf-id	A unique number for identifying a calling session on a gateway, where a session is closed when the calling party hangs up.
17	subscriber	T1/Channel Associated Signalling (CAS) or E1/R2 signal information about subscriber. Examples: Subscriber or Coin
18	session-protocol	Session protocol being used, such as SIP or H.323. Always equal to <code>SIP</code> for SIP or <code>Cisco</code> for H.323.
19	remote-media-address	Remote-media gateway IP address.
20	in-trunkgroup-label	The trunk group label associated with the group of voice ports from which the incoming TDM call arrived on the gateway.
21	in-carrier-id	Carrier ID of the trunk group through which the call arrived or the partnering voice service provider identifier of the incoming VoIP call.
22	gw-rxd-cdn	Called number as received by the gateway in the incoming signalling message before any translation rules are applied.
23	gk-xlated-cdn	Present only if the session target is 'ras' (gatekeeper routed calls).
24	gw-final-xlated-cdn	Called number to be sent out of the gateway.
25	gk-xlated-cgn	Present only if the session target is 'ras' (gatekeeper routed calls).
26	outgoing-area	Gatekeeper identifier, or the destination zone or area, of the outgoing VoIP call.
27	release-source	Indicates whether a call was released by the calling party, called party, or an internal or external source. Integer with valid values of 1 through 12, inclusive.
28	h323-conf-id	A unique call identifier generated by the gateway. Used to identify the separate billable events (calls) within a single calling session.

Field No.	Name	Description
29	h323-call-origin	Indicates the origin of the call relative to the gateway. Valid values are <code>originate</code> (initiating) and <code>answer</code> (terminating).
30	h323-call-type	Indicates call leg type. Possible values are <code>telephony</code> and <code>VoIP</code> .
31	h323-remote-address	Indicates the IP address of the remote gateway.
32	h323-setup-time	Indicates the setup time for this connection in GMT.
33	h323-connect-time	<i>Note: This field is only present in records where Acct-Status-Type equals Stop.</i> Indicates the connection time for this call leg in GMT.
34	h323-disconnect-time	<i>Note: This field is only present in records where Acct-Status-Type equals Stop.</i> Indicates the time this call leg was disconnected in GMT.
35	h323-disconnect-cause	<i>Note: This field is only present in records where Acct-Status-Type equals Stop.</i> The ISDN (Q.931) cause code associated with disconnecting the call. See <i>ISDN cause codes</i> on page 413.
36	h323-gw-id	The name of the underlying gateway.

***Release-source mapping***

The values for the Release-Source VSA are given in Table 37.

**Table 37. Release-Source Value Descriptions**

Value	Description
1	Calling party located in PSTN
2	Calling party located in VoIP network
3	Called party located in PSTN
4	Called party located in VoIP network
5	Internal release in POTS leg
6	Internal release in VOIP leg
7	Internal call-control application (Tcl or VoiceXML script)
8	Internal release in VoIP AAA
9	Console command line (CLI or MML)
10	External RADIUS server
11	External network management application
12	External call control agent

**Example 12.2(11)T record**

Below is a sample 12.2(11)T CDR, with the alternate fields in underlined blue for readability:

```
6001,7812345678,Stop,0,Local,41c,0,Login-
User,0.0.0.0,0,Async,msw,0x,,0x000000090202,h323-incoming-conf-
id=000653dc-3ede0744-70091657-
1bcb37d1@172.16.1.171,subscriber=Subscriber,session-protocol=SIP,remote-
media-address=172.16.1.171,in-trunkgroup-label=,in-carrier-id=gw-rxd-
cdn=6001,gk-xlated-cdn=6001,gw-final-xlated-cdn=6001,gk-xlated-
cgn=outgoing-area=,release-source=2,h323-conf-id=000653dc-3ede0744-
70091657-1bcb37d1@172.16.1.171,h323-call-origin=answer,h323-call-
type=VoIP,h323-remote-address=172.16.1.171,h323-setup-time=11:22:12.003
EDT Tue Aug 09 2005,h323-connect-time=11:22:31.768 EDT Tue Aug 09 2005,h323-
disconnect-time=11:22:31.768 EDT Tue Aug 09 2005,h323-disconnect-
cause=7f,h323-gw-id=csPhone
```

### ***Accounting caveats***

- RADIUS accounting messages are not real-time. Delays in accounting message propagation and processing can cause revenue leakage.

## 3GPP Rx Interface

*Note: The 3GPP Rx interface to external policy servers is an optional licensed feature. This licensed feature is available with both SBC and MSX product packages.*

### Introduction

MSX acts as a Proxy - Call Session Control Function (P-CSCF) in the 3GPP Rx architecture. In this role, the MSX provides SBC functionality and associated session services. The MSX can also act as an application manager to interface with a third party multimedia policy server for retrieving QoS policies in the cable network for multimedia session flows.

### MSX Configuration Commands

To begin using the 3GPP Rx Interface you must first create a dedicated realm for communication between the MSX server and the policy server. See *Chapter 11, iServer Realms* for instructions on creating a realm.

After creating the dedicated realm, run the following global command and answer the questions as they appear on the command line.

```
# nxconfig.pl -e enable-pcmm -v 1
enable-pcmm [0]:
Enter Local Diameter Realm for Rx Interface: [netoids.com]
Enter Local Diameter Identity for Rx Interface: []
Enter Local Diameter IP Address for Rx Interface: [0.0.0.0]
Enter Peer's Diameter Realm for Rx Interface: [netoids.com]
Enter Peer's Diameter IP Address for Rx Interface: []
Enter Peer's Diameter PortNo for Rx Interface: [1812]
Notice: iServer restart required! Restart now (y/n) [n]
Warning: Manual restart for iServer required to reflect the changes
made!
```

**Table 13. 3GPP Initial Configuration Parameters**

Question	Description	Initial Value
Local Diameter Realm	The Diameter administrative domain for the MSX server. Enter a name for the Diameter realm.	netoids.com
Local Diameter Identity	Enter the MSX host identity for the Diameter connection.	empty
Local Diameter IP Address	Enter the IP address of the dedicated realm created on the MSX server running Diameter for communication between the MSX and policy servers.	0.0.0.0
Peer's Diameter Realm	The Diameter administrative domain for the policy server. Enter a name for the Diameter realm.	netoids.com
Peer's Diameter IP Address	Enter the IP address of the policy server.	empty
Peer's Diameter Port Number	Enter the port on the policy server configured for Diameter communication.	1812
iServer Restart	You can either automatically restart the server or you can manually restart it at a later time. However, the server must be restarted for the configuration to take effect.	y/n

Use the following command to give media authorization for all calls that originate or terminate on endpoints in the dedicated realm created for the 3GPP Rx interface:

```
cli realm edit realm name mediaauth <(disabled|enforced|besteffort)>
```

The default value `disabled` does not allow requests to be sent to the policy server for call authorization. When media is authorized to send requests to the policy server, the MSX transfers the calls and upon not receiving confirmation still processes the call, performing a best effort authentication. However, if the MSX receives no authentication from the policy server, it drops the call.

## Licensing

See *Chapter 5, MSX Licenses* for a description of 3GPP Rx Interface feature licensing.



## Billing and CDR Processing

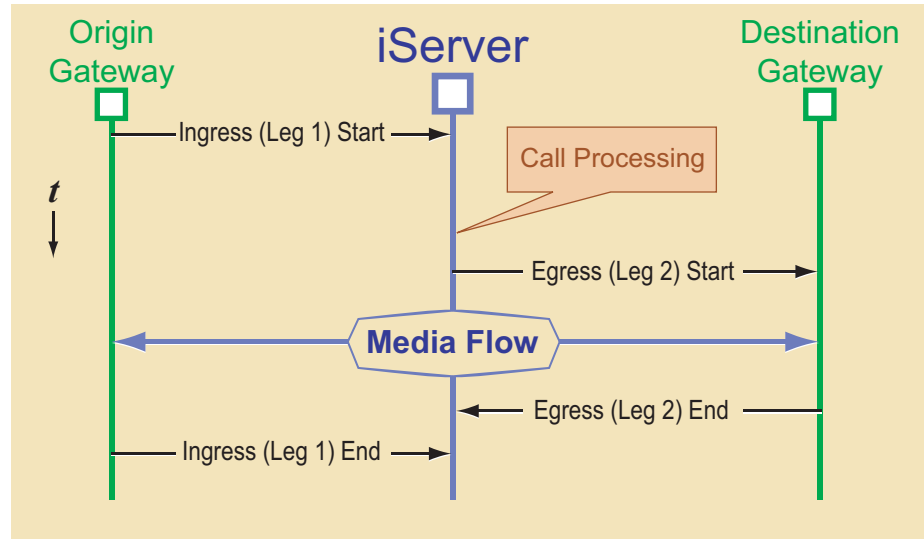
### Introduction

The MSX can log details of calls made in your network. These logs can, in turn be processed and used for billing.

*Note: While the MSX system can collect and forward billing data to other systems, MSX system is not a billing system as such. It cannot generate customer bills. A separate system is needed to turn billing data into actual bills.*

The MSX authenticates and routes each call initiated in a NexTone network. When each call or call attempt is terminated, it generates one or more Call Detail Records (CDRs) containing call specifics. In default configuration, the MSX writes one CDR into the CDR file, containing the end-call data for the ingress leg. An optional second CDR file can be generated, containing records for ingress leg start-call, egress leg start-call and egress leg end-call. Through iView, the administrator specifies which records are written to their respective log files. Figure 72, CDR-Related Call Flow, illustrates these “call legs”.

Figure 72. CDR-Related Call Flow



**Note:** *iView* uses the term “Leg 1” for the ingress leg, and “Leg 2” for the egress leg.

Once recording of ingress end-call records (traditional CDRs) is enabled, any or all of the other record types can be enabled also (that is, traditional CDRs are required, but each of the other three CDR types is optional—once the traditional ones are enabled).

Each of the two CDR file types has a unique filename suffix, and each suffix also differs depending on whether the file is open or closed. For example, the file for traditional CDRs ends with the letters “CDR”, but when it is open for writing, its last three letters are “CDT” (think of *T* as meaning *temporary*). The internal format of CDR and CDT files is identical. Table 14 details each file type and suffix.

**Table 14. Normal CDR File Suffixes by Type**

Open File	Closed File	Events Captured
CDT	CDR	Ingress end-call records
CTT	CTR	Ingress start-call, egress start-call and egress end-call records

The MSX can be configured to put CDRs for route-advance (or *hunt*) calls into the CDR files. The default is to *not* write route advance records into the CDR files, but rather only one CDR, for the last call setup attempt. Note that iView uses the term *hunt* for route-advance calls. See *Hunt CDRs* on page 428 for more information.

The MSX also logs CDRs for calls active at an MSX system shutdown. To preserve the accuracy of the CDR, the MSX automatically disconnects the call if either the calling or the called endpoint ceases to operate in the midst of a call. Such CDRs are known as *shutdown* CDRs, and contain `shutdown` in field 15.

## NexTone-format CDRs

The MSX saves CDRs in a format modeled after MIND CTI<sup>1</sup>. It is a semicolon-separated ASCII flat file format that can be imported into a billing system, or into any major package supporting CSV files.

An MSX CDR looks like the example shown in Figure 73:

**Figure 73. Sample CDR Data**

```
2003-12-16 17:10:09;1071612609;000:00:18;209.219.79.20;11089;
208.158.7.198;;;6644912112;696644912112;IV;01;N;;;12345;;;49;ee42c7001e
811cc8140fa0404badd4;000:00:10;NexTone-2600-Support;0;PopTel-
SG;0;16;6644912112;;;conn-tx#na;12345;18;;
h323;end1;1;2;;;17.852;EST;MSC-
2;6644912112;0;;CustA_realm;VndrB_realm;call_hunt_ingress_route;323gen_2;t
t;1;1
```

1. Please note that while MSX CDRs are *modeled after* MIND CTI, they do not strictly conform to that standard.

The fields that sequentially appear in each CDR are listed in tables 15 through 18. In these tables, any field listed as “Not currently used” appears as two adjacent semi-colons in the CDR file, representing a “null” field.

**Table 15. General CDR Fields**

No.	Field Name	Type/Size <sup>a</sup>	Description	Remarks
1	start-time	YYYY-MM-DD HH:MM:SS	Starting date and time of call in the local gateway	HH is between 00 and 23
2	start-time	uint32	Gateway starting time of the call in seconds since January 1, 1970 (GMT + Bias)	
3	call-duration	HH:MM:SS	Duration of call after the connection was established	
4	call-source	A.B.C.D string, max 15 chars	IP address of the originator gateway	
5	call-source-q931sig-port	uint32	Signaling port for the call	
6	call-dest	A.B.C.D string, max 15 chars	IP address of the terminating gateway	
7*	Terminator line	Integer	Line number seized by the outgoing call at the terminating gateway	Not Currently Used
8	call-source-custid	Alphanumeric string, max 24 chars	Code identifying the user originating the call	The value entered in iView's Customer ID field for the originating endpoint.
9	called-party-on-dest	string, max 64 chars	Called number in E.164 format	Number as out-pulsed to the call terminator
10	called-party-from-src	string, max 64 chars	Called number as dialed by the user	Number as sent from the ingress gateway
11	call-type	string, max 2 chars	Call type description	Call type codes are detailed in <i>Call type</i> on page 408
12*				Not Currently Used

No.	Field Name	Type/Size <sup>a</sup>	Description	Remarks
13	disconnect-error-type	string, one char.	Call disconnect reason	Call disconnect codes are detailed in Table 17, <i>Call Disconnect CDR Field (field 13)</i> on page 408.
14	call-error	uint32	Error code of the call	Error type codes are detailed in Table 18, <i>Error Type CDR Field (field 14)</i> on page 409
15	call-error	string, max 30 chars	Error code text description	
16*	(fax pages)	integer	Number of faxed pages	Not Currently Used
17*	(fax priority)	0 - Real time 1-99 – Store & Forward	Priority of fax transfer	Not Currently Used
18	ani	string, max 64 chars	Calling number identification	Used, if present from remote gateway
19*	(dnis)	N/A	Number of the destination party	No Longer Used; superseded by fields 9, 10, 31 & 50
20*	(# bytes sent)	Integer	Number of bytes sent	Not Currently Used
21*	(# bytes received)	Integer	Number of bytes received	Not Currently Used
22	cdr-seq-no	uint32	Sequence number of the CDR record	System-generated, modulo 32768
23*	(local GW stop date and time)	YYYY-MM-DD HH:MM:SS	Stopping date and time of the call in the local GW	Not Currently Used
24	callid	string, max 64 chars	Unique Call ID	See <i>Call ID</i> on page 408.
25	call-hold-time	00:00:00	The complete time during which the call occupies network resources, until it is either abandoned or connected.	Time between: <ul style="list-style-type: none"> <li>• (H.323) <i>Setup</i> and a <i>Connect</i> or <i>Release Complete</i></li> <li>• (SIP) <i>INVITE</i> and a <i>200 OK</i> or <i>BYE</i>.</li> </ul>

No.	Field Name	Type/Size <sup>a</sup>	Description	Remarks
26	call-source-regid	string	Registration ID of the call originating gateway	
27	call-source-uport	uint32	Originating gateway port number	
28	call-dest-regid	string	Registration ID of the destination gateway	
29	call-dest-uport	uint32	Port number of the call at the destination	
30	isdn-cause-code	uint32	ISDN release cause code for call leg1; range 1 to 127.	See Table 19. Zero indicates an unknown or unspecified cause code. In route-advance (hunt) CDRs, this field is blank.  See field 60 for the equivalent field for leg2.  For details, see <i>Mapping-related CDR fields</i> on page 525.
31	called-party-after-src-calling-plan	string, max 64 chars	Called number after applying source call plan	Called number after applying the call plan of the source endpoint.
32	call-error-dest	uint32	Error type codes are detailed in Table 18 on page 409	The call was disconnected on the egress leg (dest) if this error value is present.
33	call-error-dest	string	Text description of error in field 32	The call was disconnected on the egress leg (leg2) if this error value is present.
34	call-error-event-str	string (will always be a string but function can change)	Used for NexTone Debug only. The function of this field can change without notice and should only be used by NexTone engineers.	This field represents the last event in the call setup phase that was recorded for the call legs.

No.	Field Name	Type/Size <sup>a</sup>	Description	Remarks
35	new-ani	string max 64 characters	Calling number of the user after Call Plan applied.	Translated Calling Number after applying the Call Plan.
36	call-duration	uint32	Duration of call after the connection was established.	Seconds
37	incoming-leg-callid	40-character string	Either SIP Call ID or H.323 Call ID of ingress leg.	Empty for start1 and end1 CDRs
38	protocol	string	Either “sip” or “h323”	
39	cdr-type	interim string	Either start1, start2, end1, end2, hunt, or intermediate.	Hunt CDRs are written to the end2 leg file (.CTT/.CTR)
40	hunting-attempts	uint32	Call attempt # of current call (while hunting)	
41	caller-trunk-group <sup>b</sup>	string, 24 chars. max.	Trunk group of ingress call leg (H.323 calls only).	The dest TG sent from originating endpoint to the MSX
42	call-pdd	uint32	Post dial delay from SETUP to ALERTING or Progress Indicator	Milliseconds
43	h323-dest-ras-error	uint32	RAS error return code	See Table 20, “Supported RAS Reason Codes (field #43)” for details
44	h323-dest-h225-error	uint32	H.225 error return code	See Table 21, “Supported H.225 Error Codes (field #44)” for details
45	sip-dest-respcode	uint32	Response code received from SIP leg 2, if that leg was SIP and call did not connect successfully.	For details, see <i>Mapping-related CDR fields</i> on page 525.
46	dest-trunk-group	string	Name of trunk group to which call was routed	The dest. TG sent from the MSX to the dest. endpoint

No.	Field Name	Type/Size <sup>a</sup>	Description	Remarks
47	cal-duration-fractional	s.mmm	The call's duration in seconds, to thousandths	
48	timezone	string	Local time zone of the session controller	
49	msw-name	string	The SC host name from <i>servercfg</i> , or Unix hostname if blank in <i>servercfg</i>	This is also the sipserver name for SIP calls
50	called-party-after-transit-route	string	The result if a transit route has been applied to called party # (or blank)	The called party number after any transit route application (blank if none applied)
51	called-party-on-dest-num-type	uint32	The called party number type for CDR field 9	
52	called-party-from-src-num-type	uint32	The called party number type for CDR field 10	
53	call-source-realm-name	string	The name of the realm from which the call was placed	
54	call-dest-realm-name	string	The name of the realm to which the call was placed	
55	call-dest-cname	string	The egress route used to complete the call	
56	call-dest-custid	string	The egress endpoint's customer id	
57	call-zone-data	string	Zone for the ingress endpoint	
58	calling-party-on-dest-num-type	uint32	The calling party number type for CDR field 18	This is the type sent to the egress endpoint
59	calling-party-from-src-num-type	uint32	The calling party number type for CDR field 18	This is the type received from the ingress endpoint



No.	Field Name	Type/Size <sup>a</sup>	Description	Remarks
60	original-isdn-cause-code	uint32	Original endpoint-generated ISDN release cause code (leg 2), range 1 to 127.	See Table 19. Zero indicates an unknown or unspecified cause code. This field is only populated if cause code mapping is enabled. It contains the <i>un</i> -mapped code (field 30 contains the mapped code).  For additional details, see <i>Mapping-related CDR fields</i> on page 525.
61	packets-received-on-src-leg	uint32	Number of packets received on the source leg (ingress)	
62	packets-lost-on-src-leg	uint32	Number of packets lost on the source leg (ingress)	
63	packets-discarded-on-src-leg	uint32	Number of packets discarded on the source leg (ingress)	
64	pdv-on-src-leg	uint32	Packet delay variation on the source leg (ingress) in milliseconds	PDV, also known as <i>jitter</i> , is an estimate of the variance of the inter-arrival times for the ingress media packets.
65	codec-on-src-leg	string	Type of coder/decoder used on the source leg (ingress) of media-routed calls	G.711, G.729, and G.723 are examples. If the MSX is unable to determine the source codec, the G.711 $\mu$ -law codec is assumed for QoS determination purposes. Contains “unknown” for calls for which the MSX did not route media.

No.	Field Name	Type/Size <sup>a</sup>	Description	Remarks
66	(latency-on-src-leg)	uint32	Average packet latency (transit time) in milliseconds as measured by the MSX on the source leg utilizing RTCP and RTCP XR	Future; not currently used.
67	rfactor-on-src-leg	uint32	The R factor is a voice quality score (integer,0-100) based on the latency, packet loss, and jitter of the call's ingress media (RTP) stream	This is the "listening" R factor, not the "conversational" R factor. See <i>R factor (CDR field 74)</i> on page 424.
68	packets-received-on-dest-leg	uint32	Number of packets received on the destination leg (egress)	
69	packets-lost-on-dest-leg	uint32	Number of packets lost on the destination leg (egress)	
70	packets-discarded-on-dest-leg	uint32	Number of packets discarded on the destination leg (egress)	
71	pdv-on-dest-leg	uint32	Packet delay variation on the destination leg (egress) in milliseconds	PDV, also known as <i>jitter</i> , is an estimate of the variance of the inter-arrival times for the egress media packets.
72	codec-on-dest-leg	string	Type of coder/decoder used on the destination leg (egress) of media-routed calls	G.711, G.729, and G.723 are examples. If the MSX is unable to determine the destination codec, the G.711 $\mu$ -law codec is assumed for QoS determination purposes. Contains "unknown" for calls for which the MSX did not route media.

No.	Field Name	Type/Size <sup>a</sup>	Description	Remarks
73	(latency-on-dest-leg)	uint32	Average packet latency (transit time) in milliseconds as measured by the MSX on the destination leg utilizing RTCP and RTCP XR	Future; not currently used.
74	rfactor-on-dest-leg	uint32	The R factor is a voice quality score (integer, 0-100) based on the latency, packet loss, and jitter of the call's egress media (RTP) stream	This is the "listening" R factor, not the "conversational" R factor. See <i>R factor (CDR field 74)</i> on page 424.
75	sip-src-respcode	uint32	SIP response code sent to the ingress point if egress call leg did not connect successfully.	For details, see <i>Mapping-related CDR fields</i> on page 525. In route-advance (hunt) CDRs, this field is blank.
76	peer-protocol	string	The protocol for the corresponding <i>other</i> call leg.	Compare with field 38, which gives the protocol for <i>this</i> leg. Valid values: "sip" and "h323"
77	src-private-ip	A.B.C.D string, max 15 chars	The media IP address of the leg 1 endpoint if it was behind a NAT (i.e., private).	
78	dest-private-ip	A.B.C.D string, max 15 chars	The media IP address of the leg 2 endpoint if it was behind a NAT (i.e., private).	
79	src-igrp-name	string	The leg 1 endpoint's iEdge group (igrp).	
80	dest-igrp-name	string	The leg 2 endpoint's iEdge group (igrp).	

No.	Field Name	Type/Size <sup>a</sup>	Description	Remarks
81	diversion-info	string	Call diversion information for SIP calls. See Diversion Header Support on page 130.	Format: realmname,username@host,reason. May contain multiple entries for multiple diversions, each separated by a pipe character (0x7c)
82	custom-contact-tag	string	The bid and ask history of an Arbinet server's <code>redirect arbinet</code> field, in its Contact header.  Each subsequent redirect adds another "ask" entry. The last message's header contains the entire history.	
83	e911-call	string	Indicates the call was an emergency call, as determined by whether the dialed number matched one of the emergency numbers configured on the MSX (such as 911).	
84	<reserved>	string	Reserved for future use	
85	<reserved>	string	Reserved for future use	
86	call-release-source	string	Indicates the point from which the call was disconnected.	Valid values are <code>source</code> , <code>destination</code> and <code>internal</code> .
87	hunt-attempts-including-LCF-tries	uint32	Indicates all hunt attempts made, regardless of reason.	
* Fields marked as "not currently used" appear as two adjacent semi-colons in the CDR.				

a. "uint32" is a 32-bit unsigned integer field.

b. Prior to release 2.06, this field was known as *Ingress trunk group*.

**Examples:**

Table 16 shows an example of CDR fields:

**Table 16. Example of CDR Fields**

Field No.	Value
1	2004-04-20 16:54:49
2	1082494489
3	000:00:00
4	10.0.0.201
5	33240
6	10.0.0.202
7	
8	(;;) <sup>a</sup>
9	6670000000
10	6670000000
11	IV
12	01
13	E
14	1020
15	dest-relcomp
16	(;;)
17	(;;)
18	555
19	(;;)
20	(;;)
21	(;;)
22	209

Field No.	Value
23	(;;)
24	021da7144f00001029f45634343434ef
25	000:00:00
26	callgen1
27	0
28	callgen2-h323
29	0
30	115
31	6670000000
32	1020
33	dest-relcomp
34	proc-tx#proc-rx
35	555
36	0
37	(;;)
38	h323
39	end1
40	1
41	(;;)
42	0
43	(;;)
44	12
45	(;;)
46	trunk-isdn
47	0.000
48	EDT

Field No.	Value
49	msw5
50	(;;)
51	0
52	0
53	CustA_realm
54	VndrB_realm
55	call_hunt_ingress_route
56	323gen_2
57	test
58	1
59	1

- a. Two semicolons within parentheses indicates a field with no value, which simply appears in the CDR as two adjacent semicolons.

Following are examples of two CDRs:

**A good SIP call:**

```
2003-12-31
15:17:16;1072901836;000:00:05;192.168.16.37;0;192.168.16.199;;;1234567;12
34567;IV;01;N;;;3017758547;;;122;;1A9347E1-D149-432D-8895-
E94BB4BB5140@192.168.16.198;000:00:17;JohnPC;0;John's SIP
phone;0;;1234567;;ack-rx#ack-
tx;3017758547;5;;sip;endl;1;0;;;5.195;EST;william;;0;InRealmA;OutRealm
B;call_hunt_ingress_route;323gen_2;test;1;1
```

**An abandoned call:**

```
2002-07-25 02:07:05;1027577225;000:00:00;
205.113.13.166;11003;64.33.28.16;;935401515;8080011935401515;IV;01;A;2;
abandoned;;;
cf710dc900000001000c71b086b19d30;000:00:13;CarrierA;0;CarrierB;0;3;55#011
935401515;;setup-
rx#na;3015541694;0;InRealmA;OutRealmB;call_hunt_ingress_route;323gen_2;te
st;1;1
```

**Call ID**

Field 24, `callid`, is populated as follows:

- For H.323 calls, it is the value transmitted in the Conference ID field
- For SIP calls, it is the value transmitted in the `CallID` header.

In Leg2 CDRs, field 37 (incoming-leg-callid), records the `callid` of Leg1 (for subsequent correlation).

**Call type**

The call type field contains a two-character code. The first character identifies the call direction (always `I`, for *incoming*), and the second character describes the call type (`V` for voice, or `F` for fax). Every CDR has a call type; `IV` for *incoming voice*, or `IF` for *incoming fax*.

**Table 17. Call Disconnect CDR Field (field 13)**

This field describes the principal causes for a terminated call and if the call will be billed.

Value	Description	Remark
N	Normal	Will be billed
A	Abandoned	Will not be billed



Value	Description	Remark
B	Busy	Will not be billed
E	Error	Will not be billed

**Table 18. Error Type CDR Field (field 14)**

This field describes the error that resulted in the call being terminated.

Value	Type	Description / Possible Cause
0	(normal)	The user disconnected the call.
1	busy	The destination gateway or phone is busy.
2	abandoned	The caller disconnected the call before it was answered.
3	invalid-phone	The number dialed is invalid and/or unreachable.
7	user-blocked	The call was denied because the call source check failed.
12	network-error	The connection was lost due to a network error.
13	no-route	The dialed number cannot be routed.
14	no-ports	The only destination gateway found in the database is unregistered, or ports are unavailable, or its zone mismatched with the source
15	general-error	General or undefined error.
1001	resource-unavailable	Either the license (Vport) limit has been reached or the server is running low on memory.
1002	dest-unreach	Network level error; Immediate release complete.
1003	undefined	
1004	no-bandwidth	Inadequate-bandwidth returned by destination H.323 endpoint/gatekeeper.
1005	h245-error	H.245 error.
1006	incomp-addr	H.323 Incomplete address error.
1007	local-disconnect	Internal error caused local disconnect.
1008	h323-internal	H.323 stack failure.

Value	Type	Description / Possible Cause
1009	h323-proto	Protocol level failure.
1010	no-call-handle	Internal failure.
1011	gw-resource-unavailable	There are no Vports available on the gateway. This could be attributable to any of the following ISDN cause codes: 34,38, 41, 42, or 47. See CDR field # 30 or the exact code, and Table 19 for the description of the particular code for this call.
1012	fce-error-setup	Call setup failed on the Firewall Control Entity.
1013	fce-error	Internal firewall control error.
1014	no-vports	No licenses are available for the call.
1015	hairpin	Destination gateway is the same as the source gateway.
1016	shutdown	The call was disconnected due to MSX shutdown.
1017	disconnect-unreach	The call could not be connected as the destination number was unreachable.
1018	temporarily-unavailable	Call rejected by destination device. MSX internally returns this when endpoint is marked DND or gets into some kind of forwarding error or SIP 3xx loop.
1019	switchover	MSX switched to a redundant MSX and calls on this MSX were dropped because of that.
1020	dest-relcomp	Cause code is not one of those listed in this document. See the CDR cause code field for more information.
1021	fce-no-vports	MSX ran out of FCE vports. If necessary, get more licenses for FCE vports
1022	h323-maxcalls	Generated internally by MSX when max calls limit is reached for H.323
1023	msw-invalid-epid	MSX is not properly registered with the Master Gatekeeper
1024	msw-routecalltogk	MSX is not properly registered with the Master Gatekeeper
1025	msw-caller-not-registered	MSX is not properly registered with the Master Gatekeeper
1026	user-blocked-at-dest	User was blocked by called gateway on egress side and not by MSX

Value	Type	Description / Possible Cause
1027	no-route-at-dest	Egress gateway or gatekeeper replied with no route for called number.
1028	dest-timeout	MSX attempted to route call to destination, but timed out due to no response from destination
1029	dest-gone	Gatekeeper responds that the destination number has changed
1030	reject-route	The MSX has a reject route for this call
5300 <sup>a</sup>	sip-redirect	Used by the MSX only when leg2 CDRs are turned on.
5401	401 authorization required	The SIP call is not completed since it has been challenged.
5403	403 forbidden	Call authentication failed.
5407	407 proxy authorization required	The SIP call is not completed since it has been challenged.
5500	500 internal error	SIP stack failure.
5501	501 not implemented	The requested feature is not implemented.
5503	503 service unavailable	Internal error.

a. The 5000-series codes are for errors returned by SIP destinations

### **Examples:**

**Note:** *The CDRs shown in these examples do not show all CDR fields, but do show all those up to and including the ones pertinent to the example being given.*

The following sample CDR is for a *successful* call (non-zero duration):

```
2002-01-09 12:39:20;1010597960;000:17:07;192.168.1.207;25070;
192.168.1.46;;;011528183414833;528183414833;IV;01;N
;;;;;;;;;;428001910000000100004d8a7e54f52e;000:00:11;res;0;emh;1;7;55#01
1935401515;;;setup-
rx#na;3015541694;0;InRealmA;OutRealmB;call_hunt_ingress_route;323gen_2;te
st;1;1
```

Shown below is a sample CDR of a terminated call. The call was terminated due to the *destination being unreachable*.

```
2002-01-09 12:57:06;1010599026;000:00:00;192.168.1.207;25299;
192.168.1.78;;;5214532652;5214532652;IV;01;E;
1002(dest-unreach);;;;;;;;428001910000000100004e82c44565ab;
000:00:01;kes;0;emh;1;99;55#011935401515;;;setup-
rx#na;3015541694;0;InRealmA;OutRealmB;call_hunt_ingress_route;323gen_2;te
st;1;1
```

Shown below is a sample CDR of an *abandoned* call. The call was disconnected since the caller disconnected the call before it was answered:

```
2002-01-09 12:56:18;1010598978;000:00:00;192.168.1.207;
25286;192.168.1.46;;;011527222003927;527222003927;IV;
01;A;2(abandoned);;;;;;;;428001910000000100004e77ae77ade5;
000:00:45;kes;0;emh;1;112;55#011935401515;;;setup-
rx#na;3015541694;0;InRealmA;OutRealmB;call_hunt_ingress_route;323gen_2;te
st;1;1
```

Shown below is a sample CDR of a *disconnected* call. The call was disconnected since the destination gateway or phone was busy:

```
2002-01-09 15:44:57;1010609097;000:00:00;208.248.243.54;
19037;64.132.31.26;;;011527222707118;527222707118;IV;01;
B;1(busy);;;;;;;;4280019100000001000057ab89a9bc2c;000:00:04;kes;0;emh;1;5
;55#011935401515;;;setup-
rx#na;3015541694;0;InRealmA;OutRealmB;call_hunt_ingress_route;323gen_2;te
st;1;1
```

Shown below is a sample CDR of a *terminated* call. The call was terminated since either the license limit was reached or the server was running low on memory:

```
2002-01-09 15:50:44;1010609444;000:00:02;64.132.31.26;41420;
66.128.1.149;;;527221036242;527221036242;IV;01;E;
1001(resource-unavailable);;;;;;;;
4280019100000001000057fc70511760;000:00:16;kes;0;emh;1;11;55#011935401515
;;;setup-
rx#na;3015541694;0;InRealmA;OutRealmB;call_hunt_ingress_route;323gen_2;te
st;1;1
```

Shown below is a sample CDR of a terminated call. The call was terminated since an internal error caused local disconnect:

```
2002-01-09 15:51:19;1010609479;000:00:02;64.132.31.26;41526;
66.128.1.149;;;527222341559;527222341559;IV;01;E;
1007;local-disconnect;;;;;;;;;
4280019100000001000058047f8759e8;000:00:17;kes;0;emh;1;21;55#011935401515
;;;setup-
rx#na;3015541694;0;InRealmA;OutRealmB;call_hunt_ingress_route;323gen_2;te
st;1;1
```

## CDR field contents

This section details selected fields within the CDR structure.

### ISDN cause codes

Table 19 lists the ISDN cause codes supported by the MSX, along with their respective details.

**Table 19. Listed ISDN Cause Codes (field #30)**

Numeric Reason Code	Text Reason Code	Text Reason	Description / Possible Cause
1	UnassignedNumber	Unallocated (unassigned) number	ISDN number was sent to the switch in the correct format; however, the number is not assigned to any destination equipment.
2	NoRouteTransit	No route to specified transit network	ISDN exchange is asked to route the call through an unrecognized intermediate network.
3	NoRouteDest	No route to destination	Call was routed through an intermediate network that does not serve the destination address.
6		Channel Unacceptable	Service quality of the specified channel is insufficient to accept the connection.
7		Call awarded and being delivered in an established channel	User is assigned an incoming call that is being connected to an already-established call channel.
16	NormalCallClearing	Normal call clearing	Normal call clearing has occurred.

<b>Numeric Reason Code</b>	<b>Text Reason Code</b>	<b>Text Reason</b>	<b>Description / Possible Cause</b>
17	UserBusy	User is busy	Called system acknowledges the connection request but is unable to accept the call because all B channels are in use.
18	UserNotResponding	No user responding	Connection cannot be completed because the destination does not respond to the call.
19	UserNoAnswer	No answer from user (user alerted)	Destination responds to the connection request but fails to complete the connection within the prescribed time. The problem is at the remote end of the connection.
20	SubscriberAbsent	Subscriber Absent	Used when a mobile station has logged off. radio contact is not obtained with a mobile station or if a personal telecommunication user is temporarily not addressable at any user-network interface.
21	CallRejected	Call Rejected	Destination is capable of accepting the call but rejected the call for an unknown reason.
22	NumberChanged	Number changed	ISDN number used to set up the call is not assigned to any system.
26	NonSelectedUser	Non selected user clearing	Destination is capable of accepting the call but rejected the call because it was not assigned to the user.
27	DestinationOutOfOrder	Destination out of order	Destination cannot be reached because the interface is not functioning correctly, and a signaling message cannot be delivered. This might be a temporary condition, but it could last for an extended period of time. For example, the remote equipment might be turned off.
28	InvalidNumberFormat	Invalid number format	Connection could be established because the destination address was presented in an unrecognizable format or because the destination address was incomplete.
29		Facility rejected	Facility requested by the user cannot be provided by the network.

Numeric Reason Code	Text Reason Code	Text Reason	Description / Possible Cause
30		Response to status enquiry	Status message was generated in direct response to the prior receipt of a status enquiry message.
31	NormalUnspecified	Normal, unspecified	Reports the occurrence of a normal event when no standard cause applies. No action required.
34	NoCircuitAvailable	No circuit/channel available	Connection cannot be established because no appropriate channel is available to take the call.
38	NetworkOutOfOrder	Network out of order	Destination cannot be reached because the network is not functioning correctly, and the condition might last for an extended period of time. An immediate reconnect attempt will probably be unsuccessful.
41	TemporaryFailure	Temporary failure	Error occurred because the network is not functioning correctly. The problem will be resolved shortly.
42	SwitchingEquipmentCongestion	Switching equipment congestion	Destination cannot be reached because the network switching equipment is temporarily overloaded.
43		Access information discarded	Network cannot provide the requested access information.
44		Requested circuit/channel not available	Remote equipment cannot provide the requested channel for an unknown reason. This might be a temporary problem.
47	NoResource	Resources unavailable, unspecified	Requested channel or service is unavailable for an unknown reason. This might be a temporary problem.
49		Quality of service unavailable	Requested quality of service cannot be provided by the network. This might be a subscription problem.
50		Requested facility not subscribed	Remote equipment supports the requested supplementary service by subscription only.

<b>Numeric Reason Code</b>	<b>Text Reason Code</b>	<b>Text Reason</b>	<b>Description / Possible Cause</b>
55	IncomingClassBarredInCUG	Incoming class barred within CUG	Although the calling party is a member of the CUG for the incoming CUG call, incoming calls are not allowed for this member of the CUG.
57	BearerCapNotAuthorized	Bearer capability not authorized	User requested a bearer capability that the network provides, but the user is not authorized to use it. This might be a subscription problem.
58		Bearer capability not presently available	Network normally provides the requested bearer capability, but it is unavailable at the present time. This might be due to a temporary network problem or to a subscription problem.
63		Service or option not available, unspecified	Network or remote equipment was unable to provide the requested service option for an unspecified reason. This might be a subscription problem.
65		Bearer capability not implemented	Network cannot provide the bearer capability requested by the user.
66		Channel type not implemented	Network or the destination equipment does not support the requested channel type.
69		Requested facility not implemented	Remote equipment does not support the requested supplementary service.
70		Only restricted digital information bearer capability is available	Network is unable to provide unrestricted digital information bearer capability.
79		Service or option not implemented, unspecified	Network or remote equipment is unable to provide the requested service option for an unspecified reason. This might be a subscription problem.
81		Invalid call reference value	Remote equipment received a call with a call reference that is not currently in use on the user-network interface.



Numeric Reason Code	Text Reason Code	Text Reason	Description / Possible Cause
82		Identified channel does not exist	Receiving equipment is requested to use a channel that is not activated on the interface for calls.
83		A suspended call exists, but this call identity does not	Network received a call resume request. The call resume request contained a Call Identify information element that indicates that the call identity is being used for a suspended call.
84		Call identity in use	Network received a call resume request. The call resume request contained a Call Identify information element that indicates that it is in use for a suspended call.
85		No call suspended	Network received a call resume request when there was not a suspended call pending. This might be a transient error that will be resolved by successive call retries.
86		Call having the requested call identity has been cleared	Network received a call resume request. The call resume request contained a Call Identify information element, which once indicated a suspended call. However, the suspended call was cleared either by timeout or by the remote user.
88		Incompatible destination	Indicates that an attempt was made to connect to non-ISDN equipment. For example, to an analog line.
91	InvalidTransit	Invalid transit network selection	ISDN exchange was asked to route the call through an unrecognized intermediate network.
95		Invalid message, unspecified	Invalid message was received, and no standard cause applies. This is usually due to a D-channel error. If this error occurs systematically, report it to your ISDN service provider.

<b>Numeric Reason Code</b>	<b>Text Reason Code</b>	<b>Text Reason</b>	<b>Description / Possible Cause</b>
96		Mandatory information element is missing	Receiving equipment received a message that did not include one of the mandatory information elements. This is usually due to a D-channel error. If this error occurs systematically, report it to your ISDN service provider.
97		Message type nonexistent or not implemented	Receiving equipment received an unrecognized message, either because the message type was invalid or because the message type was valid but not supported. The cause is due to either a problem with the remote configuration or a problem with the local D channel.
98		Message not compatible with call state or message type nonexistent or not implemented	Remote equipment received an invalid message, and no standard cause applies. This cause is due to a D-channel error. If this error occurs systematically, report it to your ISDN service provider.
99		Information element nonexistent or not implemented	Remote equipment received a message that includes information elements, which were not recognized. This is usually due to a D-channel error. If this error occurs systematically, report it to your ISDN service provider.
100		Invalid information element contents	Remote equipment received a message that includes invalid information in the information element. This is usually due to a D-channel error.
101		Message not compatible with call state	Remote equipment received an unexpected message that does not correspond to the current state of the connection. This is usually due to a D-channel error.
102		Recovery on timer expiry	Error-handling (recovery) procedure was initiated by a timer expiry. This is usually a temporary problem.

Numeric Reason Code	Text Reason Code	Text Reason	Description / Possible Cause
111		Protocol error, unspecified	Unspecified D-channel error when no other standard cause applies.
127	Interworking	Interworking, unspecified	Event occurred, but the network does not provide causes for the action that it takes. The precise problem is unknown.

### **CDR H.225 Error Cause Code Handling**

Error cause codes logged in CDRs are derived from information available to the MSX from the endpoints with which it communicates. The MSX handles H.225 error cause codes using the following rules:

1. **If a cause code is not provided by the destination side**, the MSX sends a cause code to the source side based on Table 5 of the H.225 specs. (These mappings are included in the “Corresponding Q.931/Q.850 Cause Value” column of Table 21, “Supported H.225 Error Codes (field #44)”, below.) The cause code logged in the CDRs is always the one reported by the destination.
2. **If a cause, H.225 reason, or RAS reason is received from the destination side**, it is always logged in the CDR.
3. **If an H.225 reason or RAS reason is absent; if the call uses IWF; or if an internal error causes the call to disconnect**, the MSX uses codes and data listed in Tables 21 through 19 to send troubleshooting information back to the caller. The MSX reason is also logged in the CDR.

**Table 20. Supported RAS Reason Codes (field #43)**

Numeric Reason Code	Text Reason Code	Description / Possible Cause
1	ReasonResourceUnavailable	Gatekeeper resources exhausted
2	ReasonInsufficientResources	Bandwidth is overutilized, or no entity registered with the Gatekeeper has capacity to handle a call to the requested location at the present time
7	ReasonInvalidPermission	Permission has expired

Numeric Reason Code	Text Reason Code	Description / Possible Cause
9	ReasonInvalidEndpointID	Gatekeeper does not know about this endpoint ID: endpoint must re-register
10	ReasonCallerNotRegistered	Caller must register with the gatekeeper
11	ReasonCalledPartyNotRegistered	Gatekeeper could locate the called party, but they are not registered right now
16	ReasonRouteCallToGatekeeper	Gateway does not accept direct calls
23	ReasonRequestDenied	No bandwidth available
25	ReasonSecurityDenial	Destination has rejected call because of some policy
30	ReasonQOSControlNotSupported	MSX does not use this code
31	ReasonIncompleteAddress	Dialed number incomplete; more digits needed
33	ReasonRouteCallToSCN	ARJ is directed to an ingress gateway (the ARQ was sent by a gateway and the answerCall boolean in the ARQ is FALSE)
34	ReasonAliasesInconsistent	Multiple aliases in request identify distinct people
37	ReasonExceedsCallCapacity	The gatekeeper has determined that the destination does not have the capacity to accept this call at this point in time
38	ReasonCollectDestination	Indicates that the gatekeeper is requesting that the gateway collect the final destination address, and that the serviceControl field of the ARJ indicates the prompt to be presented to the user
39	ReasonCollectPIN	The gatekeeper is requesting that the gateway collect a personal identification number or authorization code, and that the serviceControl field of the ARJ indicates the prompt to be presented to the user
40	ReasonGenericData	The request was rejected as a result of a generic element or feature

**Table 21. Supported H.225 Error Codes (field #44)**

<b>H.225 Error Code</b>	<b>ReleaseCompleteReason Code</b>	<b>Corresponding Q.931/Q.850 Cause Value<sup>a</sup></b>	<b>Description / Possible Cause</b>
1	noBandwidth	34 – No circuit/channel available	Destination Gateway has no more bandwidth left to finish this call
2	gatekeeperResources	47 – Resource unavailable	Destination Gatekeeper ran out of resources
3	unreachableDestination	3 – No route to destination	No transport path to the destination
4	destinationRejection	16 – Normal call clearing	Destination endpoint has decided not to accept the call
5	invalidRevision	88 – Incompatible destination	Destination gateway reports that something is inconsistent about handling the call - probably that it cannot take this type of call
6	noPermission	111 – Interworking, unspecified	Called party's gatekeeper rejects the call
7	unreachableGatekeeper	38 – Network out of order	Terminal cannot reach gatekeeper for ARQ
8	gatewayResources	42 – Switching equipment congestions	Gateway has no more resources to dedicate for the call
9	badFormatAddress	28 – Invalid number format	Address format of alias not supported at destination
10	adaptiveBusy	41 – Temporary Failure	Call dropped due to LAN crowding
11	inConf	17 – User busy	No address in AlternativeAddress
12	undefinedReason	31 – Normal, unspecified	Reason is not explained by one in this list
16	facilityCallDeflection	16 – Normal call clearing	Message is sent in response to a Facility message with an empty Facility IE
17	securityDenied	31 – Normal, unspecified	Incompatible security settings
18	calledPartyNotRegistered	20 — Subscriber absent	Used by destination gatekeeper when final endpoint has preGrantedARQ to bypass ARQ/ACF

H.225 Error Code	ReleaseCompleteReason Code	Corresponding Q.931/Q.850 Cause Value <sup>a</sup>	Description / Possible Cause
19	callerNotRegistered	31 – Normal, unspecified	Used by destination gatekeeper when MSX has preGrantedARQ to bypass ARQ/ACF, or MSX is not registered
22	newConnectionNeeded	47 – Resource unavailable	The Setup was not accepted on this connection, but it may be accepted on a new connection
—	nonStandardReason	127 – Interworking, unspecified	—
—	replaceWithConferenceInvite	31 – Normal, unspecified	—
—	genericDataReason	31 – Normal, unspecified	—
—	neededFeatureNotSupported	31 – Normal, unspecified	—
—	tunnelledSignallingRejected	127 – Interworking, unspecified	—

a. These ReleaseCompleteReason code-to-Cause Code mappings are used when there is no cause code coming from the egress leg.

### **Called Party Types**

Table 22 gives the Called Party *Type Code*-to-Called Party *Type* mappings for CDR fields 51 and 52.

**Table 22. Called Party Types (fields 51 and 52)**

Type Code	Type
0	Unknown
1	International
2	National
3	Network-specific
4	Subscriber
6	Abbreviated number

Type Code	Type
7	Reserved for extension

### **QoS (Quality of Service) metrics CDR fields**

Fields 61 through 74 report quantities related to QoS measurement for downstream reporting and analysis systems. Note that the MSX does not perform QoS analysis on its own.

**Note:** *MSX QoS support requires RSM Agent of minimum version 3.2 be installed and running on the MSX.*

Voice and video applications are particularly sensitive to network quality issues. VoIP quality is readily degraded by excessive packet loss, packet delay, and variation in inter-packet arrival times (jitter). QoS metrics are increasingly needed to ensure that service level agreements are met, network-related problems debugged, and appropriate routing choices made.

### **Dependencies**

The QoS metrics reporting feature requires:

- A feature license for QoS, and
- NSF-NP

### **Overview**

Call quality measurements for both the ingress and egress call legs are written to the ingress end-call CDR (also called *Leg 1 end* CDR) at call completion. Call quality measurements, along with jitter-buffer modeling, are captured in the CDR.

### **QoS terms and definitions**

#### ***Real Time Control Protocol (RTCP)***

RTCP provides control and status for the real time protocol (RTP-RFC 3550). RTP endpoints periodically broadcast control information to each other. The primary purpose for RTCP is to provide quality of service information and it is through these broadcasts that endpoints can determine packet loss, and estimate jitter.

#### ***Latency (CDR fields 66 and 73)***

Latency, or delay, is the time necessary for network packets to traverse a particular path from end to end. In VoIP, latency generally includes not just the network propagation time, but the time it takes for the endpoint devices or their gateways to code analog voice into digitally-coded packets. It is possible to have a noticeable and annoying delay, while at the same time having

acceptable voice quality; the voice is clear, just shifted in time. End-to-end Delays in excess of 200 ms can generally be perceived.

#### ***R factor (CDR field 74)***

R factor is an industry-accepted method of quantifying subjective “call quality.” It is a voice quality metric describing a call segment carried over an RTP session. It is expressed as an integer in the range 0 to 100, with a value of 94 corresponding to “toll quality” and values of 50 or less regarded as unusable.

There are two types of R factor, called *listening* R factor and *conversational* R factor. *Conversational* R factor takes into account latency, and so is only available in CDRs where latency is non-zero for a particular leg. Where latency equals zero, the R factor reported is the *listening* R factor. Because latency is not currently reported, the R factor is always the *listening* R factor.

R factor is defined in ITU-T G.107 [6] and ETSI TS 101 329-5 [3].

#### ***Jitter (CDR fields 64 & 71)***

Jitter is the variation in the inter-packet arrival times. When successive packets have similar source-to-destination transit times, jitter is said to be low. If the variation in the transit times for successive packets is large, jitter is said to be high. For instance, if the first packet takes 50 ms to go from source to destination, and the following packet takes 60 ms, there is a 10 ms variation in their inter-packet arrival times. Jitter is a leading cause for voice quality degradation. Jitter buffers are frequently used to smooth out these variations.

#### ***Jitter buffer***

In VoIP telephony, jitter buffers are used to improve voice quality by controlling and smoothing the flow of audio packets. Network packets, especially those traversing large multi-path networks, do not necessarily arrive in an even-paced stream. In fact, packets switched through a multi-path and congested network may very well arrive out of order. A jitter buffer allows these packets to be re-queued and sent to the listener or next network hop in a smooth evenly-paced stream. As queuing introduces a nominal up-front delay, jitter buffers introduce latency (delay). Jitter buffers can be configured to use a fixed or variable size. Larger jitter buffers generally introduce greater delay and smaller buffers increase packet discards. Packet discarding occurs when packets arrive too late to be assembled and sent with their peer packets.

#### ***Jitter buffer emulation***

It is possible to predict the behavior of an actual physical jitter buffer in the network, by feeding current network quality statistics into a mathematical model. The model, utilizing information such as the buffer type (fixed,



adaptive) and size, then predicts when conditions will result in packet discards and losses. These predicted events are used to estimate packet discard/loss rates.

### ***Packet loss (CDR fields 62 & 69)***

Packets may be lost in several ways while traversing a network, including being lost to corruption (faulty/marginal infrastructure), buffer overflows, router queue discards, or a link failure. Many codecs provide some insurance against packet loss by transmitting redundant voice frames or using voice modeling to fill in the blanks.

### ***Packet discard (CDR fields 63 & 70)***

Packets are generally discarded when they arrive too late to be added to a buffer voice stream, or arrive corrupted and are unusable.

## ***Configuring the MSX to generate NexTone-format CDRs***

For a discussion of this CDR format, see *NexTone-format CDRs* on page 395.

To configure the MSX to generate CDRs in MSX format:

1. Log on to the MSX.
2. Start the `nxconfig.pl` utility by entering:

```
nxconfig.pl -E
```

This utility prompts you for settings on various options.

**Note:** *In each `nxconfig.pl` prompt, the current setting appears within square brackets following the parameter name. Pressing <Enter> keeps that setting and moves to the next parameter; typing a new value replaces the current setting.*

3. Press <Enter> several times, until the following appears

```
Billing:
```

```
-----
```

```
billingtype <a|b|c|d|e|q>
```

```
a) => none
```

```
b) => prepaid
```

```
c) => ciscoprepaid
```

```
d) => postpaid
```

```
q) => quit
```

```
Valid values are displayed above
```

```
Select your choice:
```

To change the current setting, type the letter of the new value, and press <Enter>. Note that entering `none` disables CDR recording.

4. In the next prompt, you specify the basis on which new CDR files are created.

```
cdrtype <a|b|c|d|q>
```

- a) => fixed
- b) => daily
- c) => time
- d) => seq
- q) => quit

Type the letter representing the value you want and press <Enter>. For a description of the options for this parameter, see *Setting a rule for opening new CDR log files* on page 427.

5. The next prompt asks for an interim CDR value. See *Interim CDRs* on page 428 for an explanation of interim CDRs.

```
interimcdrtimer <integer value>:
```

6. The next prompt asks for the type of events to log in CDRs. By default, end1 CDRs will always be written and that cannot be changed through this menu. You can specify additional events by entering a value, then pressing <Enter>. You will be prompted for additional values until you enter q for quit.

```
cdrevents <a|b|c|d|e|q>
```

- a) => start
- b) => start2
- c) => end2
- d) => hunt
- e) => end1 [default]
- q) => quit

7. RADIUS accounting messages can be sent to a RADIUS server, if you choose.

```
radacct <0|1>:
```

Either press <Enter>, or type a new value and press <Enter> to change it.

8. Exit the `nxconfig.pl` utility (keep pressing <Enter> until the script exits). When asked if you want to commit changes, type y. At the prompt stating that the changes may require an MSX restart, type y.

**Note:** *In-process H.323 calls, all incomplete call setups, are dropped during a restart.*

### Setting a rule for opening new CDR log files

The MSX provides four options for selecting the conditions under which new CDR log files will be opened. Any log file(s) already open will be closed when the new log file(s) are opened. The available rules are:

- Fixed—One or two new log files (.CDx<sup>2</sup>) or (.CDx and .CTx) are opened for the first call, and all CDRs from that point on are appended to the same file(s). This is the default option.
- Daily—New log file(s) are opened when the first call is made every day.
- Sequential—New log file(s) are opened when the size of either file reaches a fixed limit of 1MB.
- Time—New log file(s) are opened after a preset time interval. For instance, if you select this option for logging CDRs and specify a time interval of 15 minutes, a new log file will be opened every 15 minutes. In such a case, the CDR(s) for a call are only logged when the call is completed or terminated.<sup>3</sup>

When a call is in progress, the MSX saves temporary logs of the call in a file with a name indicating the date and time, and a .CaT<sup>4</sup> extension. On the basis selected, the MSX saves the .CaT file as a .CaR, and opens a new .CaT file for all further calls. For instance, if you configured the MSX to log CDRs daily, it does the following:

1. Opens a .CaT file(s) when the first call is made.
2. Logs “in progress” specifics for each call in this file through the day.
3. Converts this file(s) to .CaR file(s) at the end of the day.
4. Opens a new .CaT file when the first call is made the following day.

**Note:** *.CDT files can be used by NexTone personnel to debug problems. Changing the location of these files can cause logging problems, and is not recommended.*

### System time and CDT file considerations

To ensure CDR log files are properly maintained when you have selected the “Time” option for opening new CDR log files, *MSX system time should never be changed manually*. After initial system set-up, the operating system will perform this task automatically, and systems inter-communicating need to

---

2. x is either R or T, depending on the point in the logging process. (See Table 14 on page 395 for details.)

3. If selecting the “Time” option, be sure to follow the instructions given in the subsection, “System time and CDT file considerations”.

4. a is either D or T, depending on the log file creation options chosen. (See Table 14 on page 395 for details.)

have their times maintained within tolerances that cannot be maintained manually.

### ***Interim CDRs***

Fraud or hung calls may be detectable by checking call durations against a limit. The MSX provides a facility to log a special kind of CDR for calls having exceeded a pre-set duration threshold, then periodically thereafter until the call is terminated. These special records are known as “interim” CDRs, and are written to a special log file, with the name `filename.CIR`, with the same naming rules and in the same directory as the other `.CDR` log files.

Until the open file in which interim CDRs accumulate is closed, all of the interim CDRs accumulate in an open file with a `.CIT` extension. When that file is closed, the extension is automatically changed to `.CIR`. The file name is unchanged. The rule for closing a `.CIT` file is the same as for closing `.CIR` files.

### **Setting the interim CDR interval**

The limit is set by assigning a value, in minutes, to the `interimcdrtimer` attribute in `servercfg`. To set this value, log in to the MSX and enter:

```
nxconfig.pl -e interimcdrtimer -v value
```

The interval for interim CDR logging in this example is 90 minutes. This means that when a call exceeds 90 minutes’ duration, the first interim CDR for it is written to the `.CIR` file. The call is not automatically terminated at that point, but is allowed to continue.

Once this process has begun, as long as the call continues, every `interim/2` minutes thereafter (45 minutes in this example), another CDR is placed into the `.CIR` file. In this example, a CDR would be placed into the `.CIR` file at 90 minutes, then at 135 minutes, at 180 minutes, and so forth, until the call is finally torn down.

### **Interim CDR type**

Interim CDRs written to the `.CIR` file have all the usual fields, with the `cdr-type` field (field 39) containing `interim`.

## **Hunt CDRs**

The MSX can create *hunt CDRs*, which have the following characteristics:

- They are written to the file receiving `end2 leg` CDRs.

- For one call with multiple completion attempts, there will be CDRs with `cdr-type` (field 39) as follows:
  - One, with `end1`, for the final hunt attempt
  - One, with `end2`, for the final hunt attempt
  - One with `hunt` for *each* intermediate hunt attempt

## CDR data transmission via RADIUS

A selected subset of CDR data (plus addition data *not* found in CDRs) can be transmitted, to a separate machine running a RADIUS server. For details on this capability and how to set it up, see *Accounting details* on page 379 of the chapter.

# **Part 4:**

## **MSX Special Features**

## Lawful Intercept — CALEA

CALEA is the United States' *Communications Assistance to Law Enforcement Act* of 1994. This law requires communications carriers to cooperate with law enforcement agencies in intercepting calls when a warrant is issued. NexTone's iServer supports CALEA in conjunction with third-party devices (such as Xcipio™ by SS8 Networks). iServer provides access to an interception subject's call-identifying information, call content, or both. Third-party devices provide warrant provisioning and delivery of the required information to law enforcement agencies.

*Note: Countries outside the U.S. have their own names for this functionality and therefore it is often referred to by its generic equivalent, lawful intercept, or LI. CALEA is the United States' implementation.*

### Components and architecture for lawful intercept (LI)

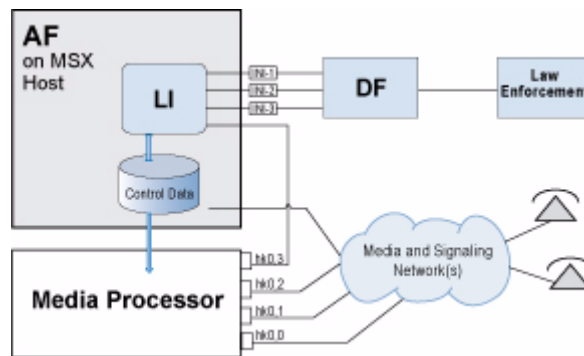
Figure 74 provides a high-level overview of the components involved in NexTone's implementation of LI. In this diagram the iServer is labeled as the AF (access function) component because it provides access to the call data requested in a warrant. The DF (delivery function) component shown in the diagram represents the DF server (such as XCipio) that delivers the requested information to law enforcement agencies.

Figure 74. LI architecture



The internal network interfaces (INIs) shown both in Figure 74 and Figure 75 are the channels for sending provisioning information, call data, and call content between the iServer and the DF server and are explained in more detail following Figure 75. The hand-over interfaces (HIs) carry the same information between the DF server and the law enforcement agency's collection function.

**Figure 75. LI network interfaces**



*Note: Although all the media processor card interfaces can support LI traffic in addition to their routing of normal media, the hk0,2 interface can be used for LI traffic only and should be configured for this purpose.*

### **The internal network interfaces for LI services**

The logical, internal network interfaces (designated INI-*n*) facilitate communication between the iServer and the DF server. Traffic for all three logical interfaces can travel across a single physical network interface as follows:

- **INI-1:** The provisioning interface, through which the DF server provisions warrants on iServer.
- **INI-2:** The call data interface, over which the iServer sends IRI (call data signaling information) to the DF server.
- **INI-3:** The call content interface, over which the iServer sends the RTP media packet stream to the DF server. The media processor card's hk0,2 interface streams the content to the iServer's LI function over an Ethernet cable to an Ethernet interface on the iServer. The iServer then forwards the media packets from the iServer to the DF server.



## Lawful intercept processing steps

The overall process for LI operations is as follows:

1. A law enforcement agency (LEA) obtains a warrant to intercept a party's calls.
2. The DF server operator enters the warrant into the DF server.
3. The DF server provisions the warrant onto the iServer. The warrant can request call data only or call data with content.
4. When the monitored party places or receives a call, the iServer forwards the call data (and if requested, the call content) to the DF server.
5. The DF server subsequently makes the collected data available to the law enforcement agency.

## How the iServer matches calls to provisioned warrants

When a DF server operator enters a warrant into the system it must include a phone number, a SIP URI, or an IP address associated with the target identified in the warrant. The DF server sends this information to iServer where it adds it to the Warrants table as a *target ID*. iServer compares header information on call INVITEs to the target IDs found in the Warrants table to determine which calls must be intercepted. Specifically:

- ♦ iServer compares SIP FROM header information to the Warrants table on ingress calls.
- ♦ iServer compares SIP TO header information to the Warrants table on egress calls.
- ♦ If iServer is performing digit manipulation on the Request URI on ingress, the FROM header is compared prior to manipulation. On egress, the TO header is compared after digit manipulation.

**Note:** *The SIP TO and FROM fields consist of two basic parts, the user part to the left of the “@” symbol and the host part to the right of the “@” symbol. For example, userpart@hostpart. During warrant matching these parts are compared to the target IDs in the Warrants table as described in the following section. Other fields found in the headers, such as display name, are ignored.*

### Matching rules for different target ID types

iServer supports three types of target IDs. iServer has specific matching rules for each type of target ID. These rules determine when iServer considers that a call matches a target ID and therefore intercepts the call. The rules for each type are described in the list that follows.

**dn** - If the target ID is of type “dn”(directory number), then the value of the field must be from 1 to 18 characters and typically represents a phone number. When iServer determines that the user part in the TO or FROM header is exactly the same as the “dn” target ID, it considers it a match. The host domain contained in the TO or FROM header is irrelevant when the target ID type is “dn.”

**url** - If the target ID is of type “url,” the values of both the user and host parts of the url must be specified in the warrant. When iServer determines that both the user part and host part found in the TO or FROM header are exactly the same as what is specified as the “url” target ID, then it considers it a match. Note that the comparison is done against the url in the URI fields only. It is not done against other fields such as the display-name in the FROM header.

**ip** - If the target ID is of type “ip,” then iServer considers as matches all calls with the specified IP address as the host part of the URI fields in the SIP TO and FROM headers, regardless of the user part.

The following table provides examples:

**Table 23. Warrant matching examples**

target ID type	target ID value iServer retrieved from header	What it matches in the Warrants table	Examples of matching SIP TO/FROM header values
dn	7035551212	7035551212 in all host part domains	7035551212@abc.com 7035551212@anything 7035551212@192.168.1.2
dn	703-555-1212	703-555-1212 in all host domains Note: Most UAs will strip the "-" and send just the digits.	703-555-1212@abc.com 703-555-1212@anything 703-555-1212@192.168.1.2
dn	1-703-555-1212	1-703-555-1212 in all host domains Note: Most UAs will strip the "-" and send just the digits.	1-703-555-1212@abc.com 1-703-555-1212@anything

**Table 23. Warrant matching examples**

target ID type	target ID value iServer retrieved from header	What it matches in the Warrants table	Examples of matching SIP TO/FROM header values
dn	+17035551212	Strip "on" (Default)17035551212 in all host domains	+17035551212@abc.com +17035551212@anything +17035551212@192.168.1.2 17035551212@abc.com 17035551212@anything 17035551212@192.168.1.2
dn	7035551212_1	7035551212_1 in all host part domains	7035551212_1@abc.com 7035551212_1@anything 7035551212_1@192.168.1.2
url	7035551212@abc.com	7035551212@abc.com	7035551212@abc.com
url	caller@abc.com	caller@abc.com	caller@abc.com
url	caller@xyz.com	caller@xyz.com	caller@xyz.com
url	+1.2.1.2.5.5.3.0.7E164@abc.com	+1.2.1.2.5.5.30.7E164@abc.com	+1.2.1.2.5.5.3.0.7E164@abc.com
ip	192.168.1.2	Anything@ IP address 192.168.1.2	7035551212@192.168.1.2 caller@192.168.1.2 5703456789@192.168.1.2

## Configuring LI support on iServer

Before you configure the iServer to support lawful intercept you must first ensure that your iServer is successfully configured (calls can be sent) including defining realms, pool, and vnets. These entities must already exist before you configure LI parameters. Then, the specific configuration to support lawful intercept consists of three basic steps:

- ✦ Verifying your license file
- ✦ Setting up the interface between the iServer and the media processing card
- ✦ Setting up communication between the iServer and the DF server

The following sections describe each of these steps.

## VERIFYING YOUR LICENSE FILE

To enable lawful intercept processing on your system, your iServer license file must include specific settings that activate this feature. Before continuing with configuration you should verify that your license file contains the necessary information. To view your license file, issue the following command:

```
nxconfig.pl -L
```

This retrieves your license information and writes it to a file, `iserverlc.xml`, in the current directory. If lawful intercept is supported on your system the file will contain entries similar to the following:

`CALL_INTERCEPT = '1'` - which indicates that LI is enabled.

`MAXWarrants='50'` - which indicates that the maximum number of warrants that can be provisioned on the iServer is 50.

`MAXIntercepts='99'` - which indicates the maximum number of concurrent calls for a target that can be intercepted is 99.

`DfServerType='SS8'` - which identifies the type of DF server you are using in your system.

License files are issued to you and are not editable. If your license file does not include the necessary information, contact NexTone Customer Support for assistance in updating your license.

## SETTING UP THE INTERFACE BETWEEN THE ISERVER AND THE MEDIA PROCESSOR CARD

You must define the interface between the media processor card and the LI process within the iServer by adding two lines to the `mdevices.xml` file. The `INTCPT-MEDIA-FWD` line defines the LI interface on the media processor card. The `INTCPT-LI` line defines the transport location where iServer's LI processing is listening for media forwarding. To add these lines:

1. Use the `-M` attribute with the `nxconfig.pl` utility to write the current contents of your `mdevices.xml` file to a new `mdevices.xml` in the same directory as follows:  

```
nxconfig.pl -M
```
2. Using a text editor, insert the `INTCPT-MEDIA-FWD` and `INTCPT-LI` lines with the appropriate values for your system. The following example

illustrates the syntax you must use for the lines. Table 24 that follows summarizes the parameters included.

```
<INTCPT-MEDIA-FWD address="10.2.0.66" mask="255.255.255.0"
gw="0.0.0.0" vlanid="0" interface="hk0,2" port="20140" />
```

```
<INTCPT-LI mac="00:0E:0C:5C:AA:DC" address="10.2.0.67"
mask="255.255.255.0" port="20000" />
```

```
</FE>
```

```
</DEV>
```

**Table 24. mDevices parameters for LI**

Attribute name	Description
<b>INTCPT-MEDIA-FWD line</b>	
address	IP address of the LI port on the media processor card.
mask	IP mask of the LI port on the media processor card.
gw	Gateway used to reach the LI process on the iServer if the media processor card and the iServer are on different subnets. A value of 0.0.0.0 means they are on the same subnet and there is no gateway. The default value is 0.0.0.0
vlanid	VLAN tag of the media processor card-LI interface. A value of 0 indicates there is no VLAN. The default value is 0.
interface	The physical port on the media processor card. This value must be set to "hk0,2" which is the default.
port	UDP port on the media processor card.
<b>INTCPT-LI line</b>	
mac	Media access control (MAC) address of the LI interface on the iServer. This should be the MAC address of the realm you are using for lawful intercept processing.
address	The IP address of the LI interface on the iServer.
mask	IP mask of the LI interface on the iServer.
port	UDP port of the LI interface on the iServer. The media processor card uses this port as the destination port and must be set to 20000.

3. After editing the `mdevices.xml` file, update the system's `mdevices` data using the `-m` option with the `nxconfig.pl` utility, for example:

```
nxconfig.pl -m -P <path>/mdevices.xml
```

where `<path>` is the path to the file. If the file is in the current directory, omit the `-P` and use `./mdevices.xml`.

4. To have changes to the `mdevices.xml` file take effect, the `iServer` must be stopped and restarted:

```
iserver all stop
```

```
iserver all start
```

## SETTING COMMUNICATION BETWEEN THE ISERVER AND THE DF SERVER

To continue configuring lawful intercept you must set some global configuration parameters that identify the DF server to the `iServer` and that establishes communications between them. These settings are added to the `servercfg` table using the `nxconfig.pl` utility (see *Global configuration using nxconfig.pl* on page 18 for information on using this utility). The following table summarizes the parameters required for the Xcipio SS8 DF server.

**Table 25. Servercfg parameters for SS8 DF servers**

Attribute Name	Description	Default
<code>nafRealmName</code>	The name of the realm to use to communicate with the DF server. The RSA of this realm and the port specified below as the Local AF transport port are used to send provisioning responses, call signaling and call content messages from the access function (AF) server, which is the <code>iServer</code> , to the DF server.  Using a separate realm for AF-DF communication is recommended.	[Null]
<code>nafLocalAfTransportPort</code>	The local transport port number for the <code>iServer</code> , for example, 20110. The <code>iServer</code> is the access function (AF) server.	0

**Table 25. Servercfg parameters for SS8 DF servers**

Attribute Name	Description	Default
nafRemoteDfHostName	The IP address or hostname of the DF server from which provisioning data (warrants and collectors) will be received. If a name is specified it should be resolvable through /etc/hosts or DNS.	[Null]
nafRemoteDfTransportPort	The IP port number corresponding to the remote DF server, above.	0
nafLocalMediaSenderUdpPort	The iServer uses the port specified here to send media to the DF server. It can be any value other than 20000, for example, 20140.	0
nafAfSerialNumber	The serial number of the iServer as it is configured on the DF server.	[Null]
nafAfModelNumber	The model number of the iServer as it is configured on the DF server.	[Null]
nafIsAuthenticationRequired	Flag indicating whether authentication is required after iServer connects to the DF server. <b>This flag should be set to 1 for SS8 DF servers.</b>	0
nafx509CertFile	Self X.509 certificate file, qualified with the full path to the file. The file should be stored so that it is not accessible to users without LI privileges if nafisSSLRequired is set to a non-zero value.	[Null]
nafx509CertParaphrase	X.509 certificate paraphrase. This is required only if nafisSSLRequired is set to a non-zero value.	[Null]
nafisSSLRequired	Flag indicating if call data and provisioning channels must be secured. <b>This should be set to 0 for SS8 DF servers.</b>	0

After you complete setting up the mdevices.xml file and the servercfg table, no additional configuration is required on the iServer to support provisioning of warrants and interception of media. Warrants are provisioned on the DF server

and therefore LI processing is initiated from the DF server side of the system (see *Lawful intercept processing steps* on page 433).

## Viewing LI-related information

For legal reasons, information on lawful intercept processing should only be accessible to users that are authorized to see it. If you are implementing lawful intercept you may also be implementing role-based user access to control who can see LI information. With role-based user access in effect, you must be assigned an “nxintercept” role in order to be able to work with LI-related information. For more information on controlling user access, see the chapter on “Role-based user access”, on page 443.

If you are authorized to do so, you can use CLI commands to view LI-related information about your system. This includes reviewing system status information and accessing information about LI processing that is stored in several iServer database tables. The warrant table includes information on the warrants provisioned on the iServer. The collector table includes information on the recipient of the data being collected. The interceptedCalls table contains information on calls that have been intercepted. The following table summarizes the available CLI commands and subcommands and the functions they perform.

**Table 26. CLI commands to monitor LI processing**

CLI command	Function performed
afserver status	Displays information on the status of the LI function within the iServer.
warrant list	Generates a list of warrants.
warrant lkup <warrantID>	Looks up a specific warrant when you supply the specific, integer warrantId associated with the warrant.
collector list	Generates a list of collectors.
collector lkup <collectorId> [collectorType]	Looks up a specific collector when you supply the specific, integer collectorId associated with the collector and, optionally, the collectorType: <ul style="list-style-type: none"> <li>• 0 - unspecified</li> <li>• 1 - CC</li> <li>• 2 - IRI</li> </ul>



**Table 26. CLI commands to monitor LI processing**

CLI command	Function performed
<code>interceptedcalls list</code> <code>[target &lt;targetid&gt;]</code>	Generates a complete list of intercepted calls or, optionally, a specific target by supplying a specific <code>targetid</code> that represents the target in the iServer.
<code>interceptedcalls lkup</code> <code>&lt;callId&gt;</code>	Looks up a specific call by supplying the specific <code>callId</code> string that represents the call in the iServer.
<code>interceptedcalls delete</code> <code>[target &lt;targetid&gt;  </code> <code>before &lt;MM/DD/YYYY&gt;  </code> <code>date &lt;MM/DD/YYYY&gt;]</code>	To delete one or more intercepted call records, where you can specify one of the following:  <code>delete</code> without any specific <code>target</code> attribute to delete all records.  <code>delete target</code> along with a specific <code>targetid</code> string to delete the record or records associated with that identifier.  <code>delete</code> with a <code>before date</code> to delete records before the date specified.  <code>delete</code> with a specific <code>date</code> attribute to delete records on the specified date.

## ENABLING DEBUG LOGS FOR LI

You can activate additional debug logging for lawful intercept processing if you find it necessary to capture greater detail about LI. To activate the debug logs, use:

```
nxconfig.pl -e debug-modincpt
```

All intercept-related logs are directed to `/var/log/intercept.log`. SSDF (signalling) message logs are logged in `/var/log/ssdf.log`.

## Using RSM Console to configure lawful intercept

You can use the graphical user interface of RSM Console to configure your system for lawful intercept processing. Instead of manually editing the `mdevices.xml` file, you can use RSM Console to define the interface between the iServer and the media processor card. You define this interface when you configure the “hk” media device on the **FCE** (Firewall Control Entity) tab of the iServer Configuration window.

As an alternative to using the `nxconfig.pl` utility, you can also set up communication between the iServer and the DF server using RSM Console. These

parameters are found on the **CALEA** tab of the iServer Configuration window. For instructions on configuring these options within RSM Console, see the RSM Console online help.

## Role-based user access

Role-based user access is a feature that allows you to control which types of operations users can perform. Users are assigned specific roles and can use only those commands for which they are authorized. For example, you can assign roles so that only certain users can make configuration changes on the iServer. When implementing lawful intercept (LI) capabilities, you can restrict access to LI-related information and log files.

### User roles

NexTone provides six different user roles which define what commands users are authorized to use. The six roles are:

- ♦ root - the user with complete system access who is responsible for initially setting up the other roles.
- ♦ sysadmin - for users who perform system administration functions such as file management and process management. This includes basic system operations such as starting and stopping the iServer.
- ♦ useradmin - for users who can manage user accounts including adding or deleting user accounts and setting passwords.
- ♦ nxadmin - for users who provision and monitor the iServer and who are authorized to write to the iServer's database.
- ♦ nxuser - for users who have only read access to the iServer database with which they can monitor status and view call data.
- ♦ nxintercept - for users who are authorized to monitor lawful intercept information.

The detailed list of commands and the type of access for each of the six roles appears in Table 27 on page 444. After implementing role-based user access, you must log into your home space directory using your NexTone iServer user

name and password. From your home directory you can run commands and work with the log files to which your role has access.

**Note:** *With the exception of the iServer-specific `cli` and `iserver` commands and the `nxconfig.pl` utility, the available commands are standard Linux and generally operate in their usual way. Role-based user access is simply a way to control which commands you can use.*

**Table 27. Privileges and commands available for each role**

Role	Privileges	Function	Available Commands		Log Access
root	Read/Write	All functions	All commands		Read/write to all log files
useradmin	Read/Write	User Management	passwd useradd	userdel usermod	no access to log files
sysadmin	Read/Write	System Management	chkconfig clear crontab date dmesg hostid hostname insserv rpm sar	shutdown top uname uptime usleep vmstat who whoami yast	Read-only access to all log files
		Process Management	kill pgrep pkill	ps pstree	
		File System Management	df du fsck fdisk	mount reiserfsck umount	

**Table 27. Privileges and commands available for each role**

Role	Privileges	Function	Available Commands		Log Access
sysadmin (continued)	Read/Write	Hardware Management	depmod hdparm hwinfo insmod lsmod modinfo modprobe rmmod	fruconfig hwreset nexbmc-config pefconfig sensor showsel	
		Security	pam_tally passwd tripwire	twadmin twprint	
		SNMP Management	nexsnmp-config snmpd	snmpget snmptrap snmpwalk	
		Networking	arp arping ifconfig ip ip6tables iptables minicom netstat nslookup	ping ping6 rarp route tethereal traceroute traceroute6 vconfig	
		Accounting	ac accton	lastcomm sa	
		File Management	basename bunzip2 bzip2 cat cksum cut diff egrep file find grep gunzip gzip	head lsof md5sum more sed sort stat strings tail tar tr wc zcat	

**Table 27. Privileges and commands available for each role**

Role	Privileges	Function	Available Commands		Log Access
nxadmin	Read/Write	iServer Management	cli egrep hostname iserver more	nxconfig.pl passwd statclient tailf uname	Read-only access to all log files except intercept.log and interceptMsg.log
nxuser	Read only	iServer Monitor	cli <sup>a</sup> egrep hostname iserver <sup>a</sup> more	nxconfig.pl <sup>a</sup> passwd statclient tailf uname	Read-only access to all log files except intercept.log and interceptMsg.log
nxintercept	Read/Write	Call Intercept Management	cli <sup>b</sup> egrep	more passwd	Read-only access to all log files

a. Although nxusers can use `cli`, `iserver`, and `nxconfig.pl`, because their access is read only, they cannot use attributes that write to the database. See Table 28 for the list of available read-only options.

b. An nxintercept user's use of `cli` is limited to specific commands related to lawful intercept. Table 29 lists the specific commands that are available.

**Table 28. Commands allowed for nxuser**

Command	subcommand (if applicable)
<b>CLI command</b>	
iedge	find lkup list cache
db	info export status hist
test	
lsalarm	
lstat	

**Table 28. Commands allowed for nxuser**

Command	subcommand (if applicable)
show	
fxs	lkup list
cp	list cache
cr	list lkup cache
stats	
ignore	
trigger	list cache
realm	list lkup cache
igrp	list lkup cache
call	cache lkup
scm	
cdc	list lkup cache
policy	list
vnet	list cache lkup
emergnum	list
blacklist	list
ratelimitbucket	list
fwzone	list

**Table 28. Commands allowed for nxuser**

Command	subcommand (if applicable)
thresholdcross	list
service-set	list
<b>iserver command</b>	
status	
version	
uptime	
<b>nxconfig.pl utility</b>	
-S (show all attribute-value pairs)	
-s <attrname> (show attribute details for <attrname>)	
-M [hostname] (get mdevices.xml file for hostname)	
-L (get license file from database)	

**Table 29. Commands allowed for nxintercept**

Command	subcommand (if applicable)
<b>CLI command</b>	
warrant	lkup list
collector	lkup list
intercepted calls	delete lkup list
afserver	status



## Setting up role-based user accounts

The *root* user account is activated when you install the iServer. Information on how to log in as the *root* user is provided by NexTone Customer Support when you initially install your system. The user accounts added when you implement role-based user access must be activated before they can be used.

### Activating default users

The role-based user access package includes one default user account for each type of role, where the user name matches the role name. There are initial accounts for users named “sysadmin,” “useradmin,” “nxadmin,” “nxuser,” and “nxintercept.” The root user activates the useradmin role by logging in and assigning a password to the default useradmin account using the `passwd` command, as follows:

```
passwd useradmin
```

You are asked to provide a password. Assigning a password to useradmin activates that user. Once the useradmin account is active, someone logged in using that default account (useradmin), or someone that has been added and assigned a useradmin role, can activate the other types of default users following a similar process of assigning passwords to the default accounts:

```
passwd [username]
```

where [username] is the name of one of the predefined roles. As before, the system prompts you to provide a password for the user. You can use these default user accounts or you can add additional users as described in the following section.

### Adding users to a role

To create additional users for any of the roles, use the `useradd` command:

```
useradd -G <nxadmin|nxuser|sysadmin|useradmin|nxintercept> <username>
```

`-G` is a mandatory argument that specifies to which role or roles this new user will belong. If you specify more than one role, separate the names with a comma. Follow the role name with the user name you are assigning to the new user. For example, to add a user named Bob as an nxadmin user:

```
useradd -G nxadmin bob
```

**Note:** *Once you implement role-based user access, only useradmin users will have access to the `useradd` command within the framework of the iServer’s role-based user access feature; therefore, useradmin users are the only ones able to add role-based users. However, this does not restrict the root user’s ability to create regular Linux users using the standard Linux `useradd` command.*

After adding the new user, assign a password using the `passwd` command:

```
passwd [username]
```

where `username` is the name assigned to the user. The system will prompt you to provide and then confirm the new password.

All user roles have access to the `passwd` command and are able to change their own password after it has been initially assigned. However, only `useradmin` users are able to change another user's password except for the root user. The root user is able to change any user's password.

### **Modifying an existing user**

If necessary you can make changes to an existing user using the `usermod` command. With this command you can change the role, or roles, to which a user is assigned, or you can change a user's username.

```
usermod [-G <role name,...>] [-l newname] <username>
```

`-G` is optional and can be followed by one or more role names that you to assign to the user. If you are specifying more than one role, separate them by commas.

`-l` is optional and can be followed by a new user name.

`<username>` is mandatory and must specify the existing user name.

For example, the following command adds the `sysadmin` role to a user, Bob, who was already assigned an `nxadmin` role:

```
usermod -G sysadmin,nxadmin bob
```

### **Deleting a user account**

You can delete an existing user account using the `userdel` command.

```
userdel <username>
```

where `<username>` is the name of the user whose account you want to delete.

## **iServer installation after role-based user access is implemented**

The `root` user is the only user authorized to upload files to the iServer. However once installation files have been loaded, a `sysadmin` user is permitted to log in and perform installation procedures. Keep in mind that `sysadmin` users can only access their authorized commands in their home directory. For this reason the normal installation procedures need to be modified slightly.

To ensure sysadmin users can reach the necessary files, when uploading installation files to iServer, the root user should place the files in the `/home/sysadmin` directory or whatever is the user's home directory. Doing so ensures that the files are accessible. The user in whose home directory the files were uploaded, whether it is the default sysadmin user or another user assigned to the sysadmin role, should log in where the installation files were uploaded and type:

```
tar -xvf install.tar
```

to extract the setup file and then type:

```
setup
```

to start installation. This differs from the standard installation step where the root user types `./setup` to start installation.

### ***Rolling back after role-based user access is implemented***

If you must roll back your system after you have implemented role-based user access, the rollback process will delete all the default user accounts and their home directories that were created when you installed. If you want to retain any of this data you should back up directories such as the `"/home/sysadmin"` directory.

In addition you should log off from any default user's account before initiating the rollback to ensure that the account is deleted correctly. If an account is not properly deleted, a root user must manually delete it by issuing the following:

```
userdel -r <username>
```

## VLAN Support

### VLANs

Virtual local area networks (VLANs) conforming to the IEEE 802.1q standard are supported by NxLinux-based MSXs incorporating NSF-NP cards. This feature is provided so that networks utilizing 802.1q will work with the MSX. The MSX's *realm-based routing* supports endpoints on private LANs without use of 802.1q VLANs, so 802.1q technology is not *required* for private network operations with an MSX. Note that this feature is not supported on releases prior to 4.x.

### VLAN high points

- Only supported on NxLinux-based MSXs with NSF-NP
- Feature license not required
- Implemented using a new MSX concept: the virtual network, or *Vnet*
- Support multiple *media* Vnets for each realm, which equates to multiple media interfaces for each defined realm.

### Elements

Logical MSX elements related to VLANs include:

- Realms
- Pools (for media only)
- MSX Port Sets (for media only)
- Vnets (new, developed for VLAN support)

As you can see, signaling and media are handled differently, with a different set of parameters and structure. They are described separately in their own sections, below.

### Realms

Realms are a means of keeping networks with the same private addresses logically separated, so that traffic originating and destined for them is correctly routed. Realms do this using a combination of dedicated signaling and media

IP addresses, and NAT'ing. See the *Realms* chapter starting on page 174 for details.

The new Vnets are configured at the realm level. Each realm may now have one 802.1q VID associated with it for its signaling IP address (that is, its RSA; only one RSA is allowed per realm), and one or more Vnets/VIDs for media, through specifying a media pool ID. The components of a Vnet are described below.

**Note:** *There are more fields associated with realms than shown in the illustrations below. The fields shown are those related to VLANs.*

### **Vnets**

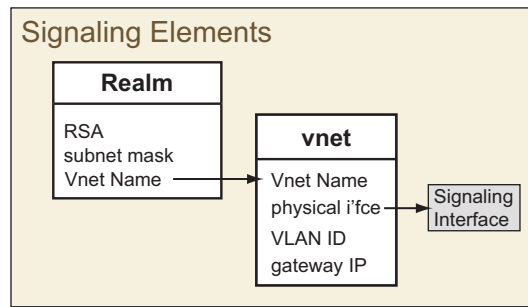
To support VLANs, the MSX includes the concept of a “virtual network,” or *Vnet*. The 802.1q VLAN standard uses a 12-bit “VLAN ID” (or *VID*) inserted into the IP frame to associate the frame with a particular VLAN. As shown in the illustrations below, the Vnet object contains a VID and a physical interface specification. Note that each Vnet object on the MSX must have a unique name, as well as unique physical interface/VID combination. Vnet names can be up to 31 alphanumeric characters long. Valid VIDs range from 1 to 4094 inclusive. Once created, a media Vnet's name should not be changed. Signaling Vnet names *cannot* be changed once they're created.

“None” is a valid selection for VID.

Vnets for signaling and media contain different information, and are stored in different places on the MSX. Signaling Vnet data is kept in a binary, non-viewable “database” file, maintained with CLI commands. Media Vnet data is kept in the `mdevices.xml` *servercfg* attribute for each machine, and is best maintained through RSM Console.

## Signaling

The relationships between the two signaling elements that relate to VLANs is depicted in the figure below. Each element is described in the sections that follow.



### Realm

On systems supporting VLANs, each realm specifies a subnet mask and signaling Vnet for it to use.

*Note: There are more fields associated with realms than shown in the illustration. The fields shown are those related to VLANs.*

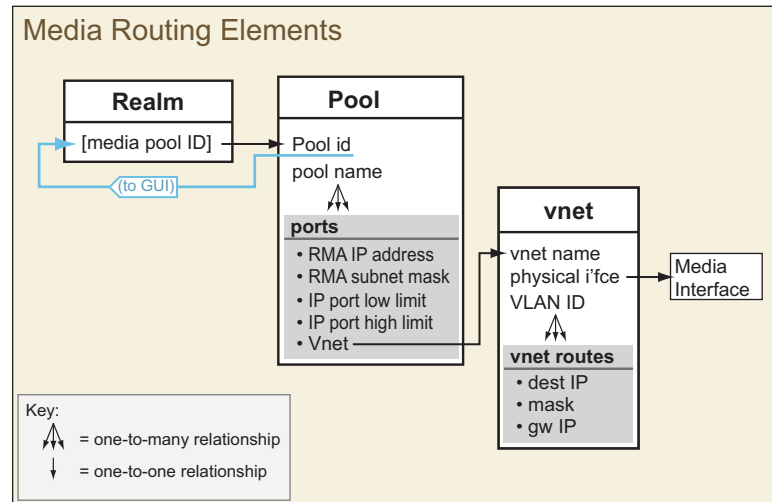
### Vnet

Every realm names a signaling Vnet, which in turn supplies:

- The physical interface that realm communicates through.
- An 802.1q VLAN ID associated with packets originating or terminating in that realm.
- The IP address of the gateway through which the MSX reaches the realm.

## Media

The relationships between the media routing elements that relate to VLANs is depicted in the figure below. Each element type is described in the sections below.



## Realms

Each MSX realm specifies a numeric Pool ID to link to its designated pool. It uses pools (groups of IP addresses and ports; see “Pools”, below) to route media through RMAs (realm media addresses). The *Realms* chapter starting on page 174 provides details about RMAs and how they work in concert with policy-based routing.

**Note:** *There are more fields associated with realms than shown in the illustration. The field shown is the one related to VLANs.*

## Pools

MSX Pools are logical, named groupings of firewall resources available for realm-based media routing. They are described in detail in the *Realms* chapter starting on page 174. On an MSX supporting VLANs, a media pool provides links to one or more port objects (see below.) Note that signaling does not use pools on Linux-based MSXs, though it did on Solaris-based MSXs. Each pool has a unique numeric ID, and a human-readable name given by the administrator.

## Ports

Media ports provide one or more IP address/subnet mask sets, with a range of IP port numbers (as a low [starting number] and high [ending number]) for that

IP address. They also name a Vnet to use for each media port. Signaling does not use ports objects, since the signaling port is specified in the MSX's configuration.

### **Vnets**

Each realm names one or more media Vnets, which in turn supply:

- The physical interface that realm communicates through
- An 802.1q VLAN ID associated with packets originating or terminating in that realm
- Optionally, one or more routing table entries, consisting of:
  - The destination IP address
  - Subnet mask
  - IP address of the gateway through which the MSX reaches the realm

### ***Element definitions***

The three elements subordinate to realms (*pools*, *port ranges*, and *Vnets*) are all defined in the `mdevices.xml` attribute in *servercfg*. Each machine has one `mdevices.xml`. Directly editing `mdevices.xml` is not recommended, and really isn't necessary, since RSM Console facilities are provided for editing `mdevices.xml`, with checks to safely maintain the contents of it.

As noted above, Vnets are only supported on Linux-based MSXs with the NSF-NP option installed.

### ***CLI operations***

Certain operations related to VLAN support can be performed through the command line-interface (CLI). Specifically, CLI commands are available for setting up and maintaining *signaling* Vnets, and configuring realms to use them. Note that pool creation on a Linux system only relates to media routing, and that all media-specific Vnet configuration work must be performed using RSM Console.

Signaling Vnets may also be set up and maintained through RSM Console, which is the preferred method, because of the safety checks it affords. *Media* Vnet setup and maintenance should only be done using RSM Console, as described in Chapter 15, "Media Services", on page 294.

The general procedure is to create the lower-level element first (the Vnet), then the higher-level element (realm), since the high-level refers to the low, and



reference cannot reliably be made to an element which has not yet been created.

### **cli vnet**

The following table lists the commands relating to signaling Vnets (for setting up *media* Vnets, you *must* use RSM Console):

*Note: The commands shown below are in standard Unix “man” format. Underlined values, such as vnetname, are the data you supply.*

**Table 30. cli Vnet Commands for Signaling Vnets**

Task	Command	Options/Notes
Create a new signaling Vnet	cli vnet add <u>vnetname</u>	31 chars. max.
Delete an existing signaling Vnet	cli vnet delete <u>vnetname</u>	
Edit existing signaling Vnet characteristics	cli vnet edit <u>vnetname</u>	ifname <u>physInterfaceName</u> vlanid <u>VID</u> [0-4094] gateway <u>gatewayIP</u> fwzone <u>fwzone_name</u> see Chapter 17, <i>Signaling Firewall Operations</i> , on page 339 for more information on the fwzone option.
Show all signaling Vnets defined	cli vnet list	Gives details
Show signaling Vnets currently in the MSX's active cache	cli vnet cache	
Show details for one defined signaling Vnet	cli vnet lkup <u>vnetname</u>	

### **cli realm edit**

The following table lists the commands used to relate signaling Vnets to realms:

**Table 31. cli realm edit Command for Signaling Vnets**

Task	Command	Options/Notes
Assign signaling Vnet(s) to a realm	<code>cli realm edit <u>realmname</u> <u>vnet-name</u> [<u>vnetname</u> ...]</code>	

### ***RSM Console operations***

This section presents the operations available for creating and maintaining signaling and media Vnets through RSM Console. Note that media VLAN support is only available through RSM Console, not CLI, but that CLI commands are available for signaling VLAN setup and maintenance.

MSX supports VLANs through several building blocks (realms, pools, Vnets, etc.), which are dependent on one another. Because of these dependent relationships, the lowest-level items are created first, then the sequentially higher-level items. The hierarchy (low-to-high) is:

1. Vnets (including optional routes for media Vnets)
2. Pools (media only; includes port ranges)
3. Realms

The procedure to modify media vnets, pools, and realms is fully documented in Chapter 15, “Media Services”, on page 294, starting at the section *Setting up media devices and media routing pools*, on page 297.

Signaling Vnets are configured in a separate part of the RSM Console interface. Procedures for configuring signaling Vnets are provided below.

### **Vnets**

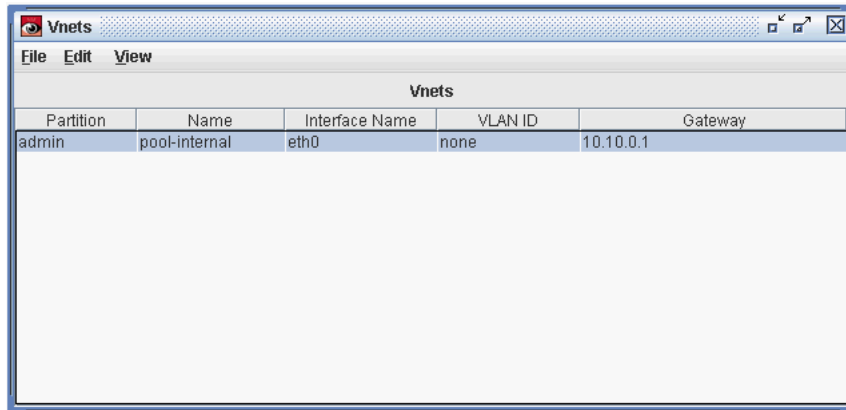
#### ***Create a Signaling Vnet***

To create a signaling Vnet

1. Start RSM Console and double-click the MSX you want to configure  
The MSX main window opens.
2. Select **Utilities** → **Vnet** from the menu bar.

The **Vnet** window appears listing any previously configured Vnets.

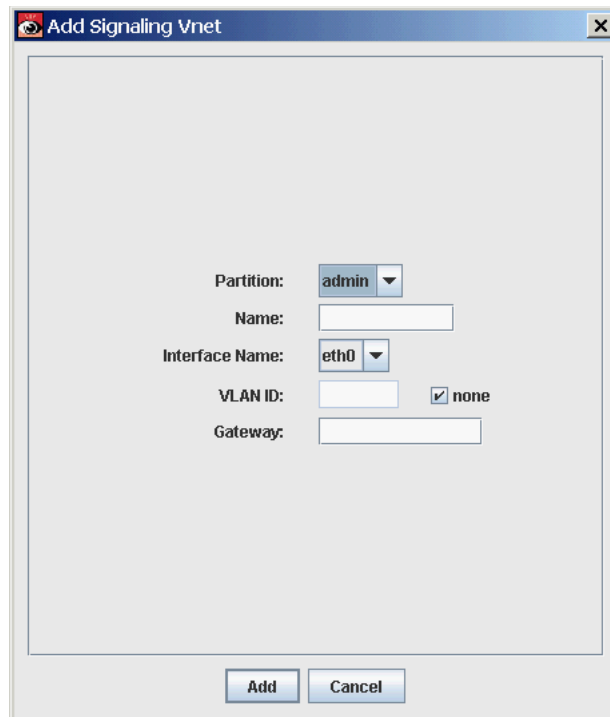
**Figure 76. Vnets Window**



3. Select **Edit->Add** from the Vnet window menu bar.

The Add Signaling Vnet dialog shown in Figure 77 appears.

**Figure 77. Add Signaling Vnet Dialog**



1. From the Partition pull-down list, select the partition to which the signaling Vnet will apply.
2. In the Name box, enter a unique, text name for the new Vnet, up to 31 characters long.
3. From the Interface Name pull-down list, select the physical interface for this signaling Vnet.
4. In the VLAN ID box, enter an 802.1q VID from 1 through 4094, or select the none box.
5. In the Gateway box, enter the IP address of the gateway through which the MSX accesses the realms that will use this Vnet.
6. Click Add to save the new Vnet, or Cancel to exit the dialog without saving your changes.

#### ***Modify a signaling Vnet***

1. Start RSM Console and double-click the MSX you want to configure  
The MSX main window opens.
2. Select **Utilities -> Vnet** from the menu bar.  
The **Vnet** window similar to the one in Figure 76, on page 459 appears listing any previously configured Vnets.

3. Right-click the Vnet to modify from the Vnet window and select **Modify Vnet**. The **Modify Signaling Vnet** window appears:

**Figure 78. Modify a Signaling Vnet**

The screenshot shows a window titled "Modify Signaling Vnet". It contains the following fields and controls:

- Partition:** A dropdown menu with "admin" selected.
- Name:** A text field containing "pool-internal".
- Interface Name:** A dropdown menu with "eth2" selected.
- VLAN ID:** A text field that is currently empty, with a checkbox labeled "none" checked to its right.
- Gateway:** A text field containing the IP address "10.10.0.1".
- Buttons:** "Modify" and "Cancel" buttons at the bottom.

4. The following fields can be modified:
  - **Partition**—specify a different partition for the signaling Vnet.
  - **Interface Name**—select a different physical interface for the signaling Vnet.
  - **VLAN ID**—edit the 802.1q VID. Use a value from 1–4094, or select the none checkbox.
  - **Gateway**—edit the IP address of the gateway through which the MSX accesses the realms that will use this Vnet.
5. Click **Modify** to save your changes, or **Cancel** to exit without saving.

#### **Delete a signaling Vnet**

To delete a signaling Vnet:

1. Start RSM Console and double-click the MSX you want to configure.

- The MSX main window opens.
2. Select **iServer->Configure** from the menu bar.  
The MSXConfiguration window opens.
  3. Select **Utilities->Vnet** from the menu bar.  
The Vnets window opens.
  4. Right-click the Vnet you want to delete and select **Delete** from the popup menu.

### **Realms**

For general information and details on creating and maintaining realms with RSM Console, see the *Realms* chapter of the *Installation and Operations Guide*. The information provided here is limited to what you will need to do with realms in order to implement VLANs.

When creating a realm, you enter a signaling Vnet name, and a media pool ID. You can also add that information after creating the realm. The basic procedure is the same in both cases, so the procedure to modify an existing realm is the only procedure presented here.

**Figure 79. The Realms Utility (partial)**

[illegible]

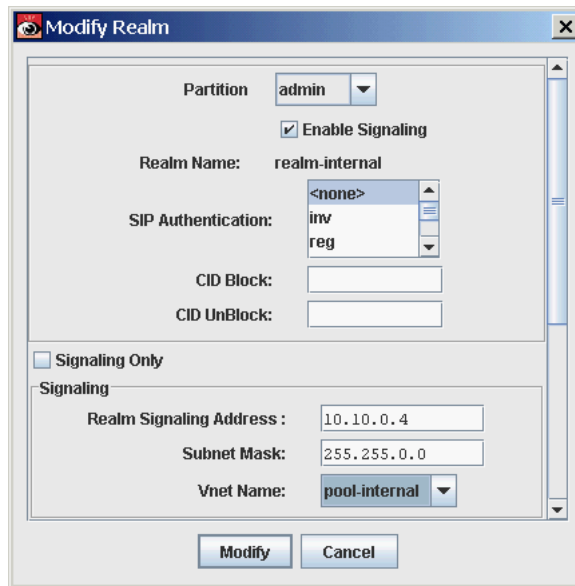
---

### Modify a realm to use a signaling Vnet

The procedure below is for assigning a signaling Vnet after the realm's initial creation.

2. Double-click the realm, or right click on it, and choose **Modify...** The **Modify Realm** window appears.

**Figure 80. Assign a Signaling Vnet to a Realm**



3. In the Signaling section, select a Vnet from the Vnet Name pull-down list.
4. Click **Modify** to save your work and close the window, or **Cancel** to close the window without saving any changes.

#### ***Modify a realm to use media Vnets***

The procedure below is for assigning a media routing Vnet after the realm's initial creation. Remember that Vnets are part of media pools when speaking of media, so you will assign a pool that has the Vnet in it, not the Vnet directly.

1. Open the **Realm** utility (**Utilities** → **Realm**), and locate the realm to modify, in the list of defined realms, as shown above.



2. Double-click the realm, or right click on it, and choose Modify... The **Modify Realm** window appears.

**Figure 81. Assigning a Media Vnet to a Realm**

The screenshot shows the 'Modify Realm' dialog box with the following settings:

- Partition:** admin
- Enable Signaling:** ☒
- Realm Name:** realm-internal
- SIP Authentication:** <none> (selected), inv, reg
- CID Block:** (empty)
- CID Unblock:** (empty)
- Vnet Name:** pool-internal
- Media Section:**
  - Media Pool ID:** 2
  - Between Realms Media Routing:** Always On
  - Within Realm Media Routing:** Always Off
- Mirror Proxy Section:**
  - Registration ID:** (empty)
  - Port:** (empty)
- Public/Private:** Private (selected)
- Buttons:** Modify, Cancel

3. In the Media section, select a Media Pool ID, from the pull-down list, that includes the Vnet(s) you want to assign to that realm.
4. Click Modify to save your work and close the dialog, or Cancel to close the dialog without saving any changes.

## Calling Plans

### What is a calling plan?

A calling plan is a set of one or more routes, identified by a single name. Calling plans can be assigned to one or many endpoint ports.

A calling plan specifies preferences and policies for incoming and outgoing calls. By assigning calling plans to one or more endpoint ports, the user controls how calls are handled.

Calling plans, routes and endpoints relate as follows:

- A calling plan can contain any number of routes
- A route can belong to any number of calling plans
- An endpoint port can have only one calling plan assigned to it.

To summarize, an endpoint has one set of routes assigned to it: those contained in *the* calling plan bound to that endpoint. The section, *Call route bindings* on page 470 describes this relationship in greater detail.

### Routes

A calling plan consists of one or more call *routes*. Each call route consists of<sup>1</sup>:

1. **Route Name.** A name unique to each route.
2. **ANI: Calling Party # and Length (or Length unspecified).** Call origination number (“caller ID”) parameters for route selection.
3. **ANI: Prefix.** Optional digits to be prepended to the “from” number passed to the egress gateway.
4. **DNIS: Called Party # and Length (or Length unspecified).** Dialed-number matching parameters for route selection.
5. **DNIS Prefix/NoDigit Strip.** Optional digits to be prepended to the number “dialed” to the egress gateway.

---

1. These parameters are set in the *Modify Calling Plan Route* panel of the *Calling Plan* utility. Table 32 provides details of these parameters.

6. **Default Route option.** Checking this number designates the route as a calling plan's "default route" that will match any dialed number. This route always matches an "empty" dialed number.
7. **Reject option.** An egress route can be defined as a "reject route." See *Reject routes* on page 482 for more information.
8. **Sticky option.** A route with this option checked is a "sticky" route. Typically used with call hunting, this means that either a call completes with this route, or the call fails. For more information, see *Sticky routes* on page 482. Endpoints can also be designated as sticky (see *Making an endpoint "sticky"* on page 78).
9. **Applied at** - This specifies where the calling plan route is applied. Options are *ingress* point (used for digit normalization), *egress* point (to determine where the call leaves the network), or in *transit* (that is, applied to *all* calls.) See *Transit routes* on page 486 for details.
10. **Template option.** If this option is selected, the route is not used for call routing, but as a basis for 'true' routes created dynamically.

Table 32 lists the parameters defining a route, and affecting how routes are applied. These parameters are detailed in the sections that follow.

**Table 32. Route-Defining Parameters**

Group	Parameter	Description	Comment
(all)	Route Name	A user-assigned unique name for a particular route	String, max. 28 characters <sup>a</sup>
ANI	Calling Party	The E.164 number from which the call was placed	If these are supplied, the route is only applied if the call-origination parameters match those specified here.
	Length   Length Unspecified	The number of ANI digits to examine when determining whether to apply the plan	
	Prefix	A number optionally prepended to the whole or part of the calling phone number	

Group	Parameter	Description	Comment
<b>DNIS</b>	Called Party Number	The DTMF digits dialed on the originating device. These digits get stripped unless the NoDigitStrip option is checked.	If these are supplied, the route is only applied if the call-destination parameters match those specified here.
	Length   Length Unspecified	The number of digits the dialed number must contain for the route to match. Can be left blank by choosing "Length Unspecified."	
	Prefix	The iServer prepends this number to the called number after it removes the Called Party Number from it. If this field is blank, the iServer does not prepend any number to the called number.	
	NoDigit Strip	Checking this prevents the matched digits in the Called Party Number from being stripped when applying the route. Checking this disables specifying a prefix.	
	Default Route	Checking this box makes this route always match an empty dialed number. See <i>DNIS default route</i> on page 501.	
	Reject	If checked, this egress route specifies an exception to a higher-level route.	
	Sticky	If checked, this route is the end of route advancing. That is, if the call cannot be completed with any endpoints bound to this route, no other routes are tried, and the call doesn't complete.	
<b>Properties</b>	Applied at	Specifies where the calling plan route is applied.	The mutually-exclusive options are <i>Ingress</i> , <i>egress</i> , and <i>transit</i> .
	Template	If checked, this route is used only as a basis for derivative routes, and not for call routing as it stands.	

a. The route name can actually contain 96 characters, but only 28 define uniqueness.

### Bulk route creation

A tool that facilitates bulk loading of call routes is available from NexTone. The routes are placed in a comma-separated text file, that is fed into the tool, called Gen\_CP, which then generates XML code for importation into the iServer's database.

The latest version of Gen\_CP can be [downloaded from NexTone](http://gencp.nexttone.com:8080/gencp-1-14/GenCP-1-14.html). The link for this is: <http://gencp.nexttone.com:8080/gencp-1-14/GenCP-1-14.html>

For details on this tool and how to use it, contact NexTone Support.

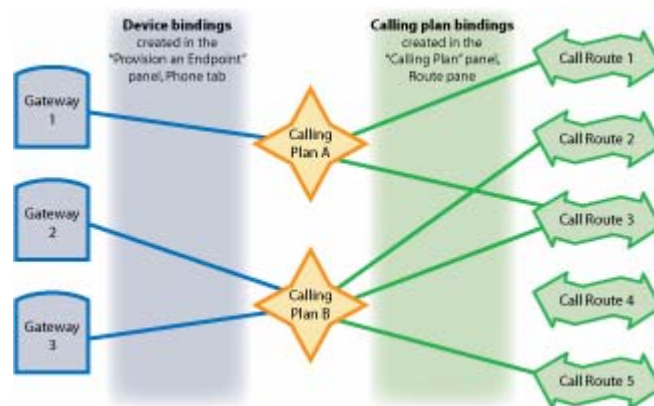
### Call route bindings

There are two types of *binding* that govern the use of each calling plan:

- *Calling plan* bindings, linking calling plans to call routes, and
- *Device* bindings, that link devices to calling plans

Figure 82 illustrates the two binding types that together implement a calling plan.

**Figure 82. Calling Plan Bindings**



Note the situations depicted, all of which are valid:

- One calling plan (A) bound to one device (GW1)
- One calling plan (B) bound to multiple devices (GW2 and GW3)
- One call route (CR1) bound to one calling plan (A)
- One call route (CR3) bound to multiple calling plans (A & B)
- A call route (CR4) that is not yet linked to a calling plan

Note also what is *not* depicted: one device bound to more than one calling plan. That is not a valid configuration, and the iServer doesn't allow it.

### **Calling plan binding**

Calling plan bindings link a call route to one or more calling plans. This binding details the rules to be applied by the iServer when applying that route to a call. Thus, a route belonging to two calling plans has two calling plan bindings, one to each plan.

The calling plan binding of a call route specifies the following attributes for the route:

- **Priority** —These are relative priorities between routes. If two or more call routes match the criteria for routing a particular call, the iServer routes that call using the route with the numerically-higher priority.
- **Route type** —If the Reject box is checked, this route is a *reject route*. See *Reject routes* on page 482 for more information.
- **Start and End Dates and Times** —When specified, these attributes indicate the point(s) in time when the call route is active.

### **Device binding**

A device binding assigns a calling plan to an endpoint port, and sets the device's call-routing priority. If multiple calling plans qualify to route a particular call, the iServer routes the call to the endpoint/port with the numerically-highest device binding priority first.

Although an endpoint cannot bind to more than one calling plan, a calling plan *can* be bound to multiple endpoints, as shown in Figure 82.

## **Call legs**

By convention, calls are said to have *legs* (that is, a call comes into a network on one *leg*, and leaves that network on another *leg*.) A call's two primary legs are loosely known by several equivalent terms. That is:

- The leg on which the call enters the network is known variously as:
  - Leg 1
  - The source leg (often abbreviated as *src*)
  - The origin leg (often abbreviated as *orig*)
  - The ingress leg
  - The incoming (or *in*) leg

- The leg on which the call leaves the network is known variously as:
  - Leg 2
  - The destination leg (often abbreviated as *dest*)
  - The egress leg
  - The outgoing (or *out*) leg

In various contexts, you will hear these terms used interchangeably. They are viewed as from the perspective of the network controlling device, in this case the iServer, not the endpoint. For example, the ingress leg is the leg from a gateway to the iServer, not the one going into the gateway.

**Note:** *Strictly speaking, not all calls have only the two basic types of leg named above. In certain situations (such as transcoding), call traffic may leave the iServer not destined for an egress point and return to the iServer before actually departing the network. In this documentation, unless otherwise noted, “legs” refer only to ingress and egress call segments, not such other, intermediate hops.*

## Creating a calling plan

For calling plans to be applied for calls at an endpoint, the calling plan information for that endpoint must be in the iServer database, entered using RSM Console's *Calling Plan* utility<sup>2</sup>. The general steps are:

1. Create a new calling plan name on the iServer
2. Create one or more call routes for the plan, or re-use existing routes, as appropriate
3. Bind route(s) to the calling plan
4. Bind one or more endpoint ports to the calling plan (from the endpoint's *Provision* or *Modify* window)

Refer to RSM Console online help for the detailed procedure to implement these steps.

### Prioritizing calling plans

When binding a call route to a calling plan, you can set a priority for the route. When routing a call, if the iServer finds two or more routes that qualify to route a particular call, it selects the route with the numerically-higher priority. You

---

2. While RSM Console is preferred, CLI commands can also be used.



can set the priority for a route as part of its calling plan binding properties on RSM Console's *Modify Calling Plan Binding* panel.

### **Setting a time of day for calling plan operation**

A route-to-calling plan binding can specify times during which the iServer applies the route. This is done using the Start and End Time boxes on RSM Console's *Modify Calling Plan Binding* panel. If specified, the iServer considers this attribute when selecting or bypassing the route for the call. Refer to RSM Console's online help for the detailed procedure.

## **Call trace route**

The Call Trace Route utility tracks the route a call takes, detailing the iServer's decision of ports to route the call at each step. This is useful for checking how a particular calling plan will handle real calls, without actually having to place test calls and trace them manually.

Refer to RSM Console's online help for details on using the Trace Route utility.

## **Calling plan operation**

The iServer routes calls using routes bound to calling plans stored in its database, applying them as follows:

1. The iServer determines the point of application (at the ingress port or egress port, or during transit (see *Transit routes* on page 486)).
2. Considering all routes in the calling plan assigned to this port, the iServer matches the called phone number and its length with the destination number and length specified in the routes. If it finds a match, it applies that route. If no route matches, the call is rejected.
3. From the called number, the iServer removes the sequence of digits that match the Called Party number specified in the route.
4. To the resultant number, the iServer prepends the new prefix number specified in the calling plan route, and dials out this new number from the port to which it is bound.

**Note:** *When routing calls, the iServer considers ALL endpoints on the network, irrespective of whether they are dynamic or static devices. It can therefore, route calls to an endpoint, even if the endpoint has not sent a "keep alive" to the iServer.*

For clarity, let us consider the following example, and illustrate each step with it:

If,

The dialed phone number is 8744066

A route is configured as follows:

The DNIS Called Party # is 874, with DNIS Length of 7

The DNIS Prefix to be inserted is 240

The route is to be applied at the ingress point

The route is configured on the source endpoint

Then for this example,

1. The iServer performs all route application at the ingress port (calling endpoint).
2. For each route in the plan, the iServer compares:
  - 2.1 the DNIS Called Party # to the first three digits of the dialed number (since the DNIS Called Party # specifies three digits), and
  - b. the DNIS Length to the total number of digits in the dialed number.

If both of these parameters match, the iServer applies this route to the dialed number.

In our example, the DNIS Called Party # parameter (874) matches the first three digits of the dialed number. Therefore, the iServer applies this ingress route.

3. The iServer removes (strips) those digits in the dialed number that match the specified DNIS Called Party # in the route.

In our example, the matching digits in the two numbers are 874. The iServer removes these digits from the dialed number, obtaining 4066.

4. Next, the iServer prepends the digits in the new DNIS Prefix to the result to get a new phone number which is then dialed out from the egress port.

In our example, the new prefix is 240. The iServer prepends this number to the result of step 3 (4066), and dials out the phone number 2404066.

**Destination number match operation**

When the iServer selects a route for a call based on the route's DNIS Called Party #, it operates as explained in the steps below. For clarity, let us consider the following example, and illustrate each step with it:

If,

The dialed phone number is *4536308*

The DNIS Called Party #/Length in the calling plan route are *453 / 7*

The DNIS Prefix is *240453*

The route is applied at any valid point: source, destination or in-transit.

Then,

1. The iServer compares the first three digits of the dialed phone number to the DNIS Called Party #. It then compares the length of the dialed number with the DNIS Length. If the digits and the lengths match, it selects this route. (If either the number or the length does not match, the iServer continues searching for a matching route.)

In our example, the DNIS Called Party #, *453*, matches with the first three digits of the dialed phone number, and the destination length 7 matches the number of digits in the dialed phone number. The iServer selects this route.

2. The iServer strips those digits in the dialed number that match the destination number.

The sequence of digits that match the destination number are *453*, so the *453* is removed from the dialed phone number, yielding just *6308*.

3. The iServer prepends the DNIS Prefix to the result.

The DNIS Prefix for this example route is *240453*, which the iServer prepends to the *6308* result from step 2. *2404536308* is dialed out.

**Null destination pattern**

You can configure a route to match any dialed number by leaving the DNIS Called Party # field blank.

For instance,

If,

The dialed number is 3355328

The DNIS Called Party # / Length is *blank*/7

The DNIS Prefix is 2404536308

Then,

1. Since the destination number in the call route is left blank, the iServer does not check for a match of numbers. However, it does check for the destination length. Hence, it uses this route for a call from any number, provided:

- It does not find an exact match in any other routes.
- The length of the dialed number equals the DNIS Length.

For instance, in this example, the iServer uses the above route for the dialed phone number 3355328 if no exact match is found in any other route, since the length of the dialed number equals the destination length of 7.

2. The iServer prepends the new prefix to the dialed phone number, without stripping any digits from it.

In this example, the iServer prepends the new prefix 2404536308 to the dialed phone number 3355328, resulting in a new number, 24045363083355328. This is now the number that is routed through the gateway.

The general rule is that matched digits are always stripped (unless the *No Digit Strip* option is checked for the route). In this case, there were no matched digits, so no stripping took place.

### ***Destination length unspecified***

A call route with an unspecified DNIS Length is selected when the first digit(s) of the DNIS Called Party # match the dialed number.

For instance:

If,

The dialed number is *3355328*

The DNIS Called Party # / Length is *335/unspecified*

The new prefix is *240453*

Then,

1. The iServer checks for a match between the DNIS Called Party # and the dialed phone number, but does not consider the DNIS Length.

For instance, in our example, the iServer uses the above route for the dialed phone number, since the DNIS Called Party # *335* matches the first three digits of the dialed phone number *3355328*.

2. Now, as in all other cases (where No Digit Strip is not enabled), the iServer removes the sequence of digits in the dialed number that match those in the DNIS Called Party #.

In our example, the sequence of digits that match the destination number are *335*. The iServer removes this sequence from the dialed phone number, and the result is *5328*.

3. The iServer then prepends the new prefix to the result of step 2's stripping of the matched digits.

In our example, the new prefix is *240453*. The iServer prepends this number to the result of step 2, *5328*. The number dialed from this gateway is *2404535328*.

## Calling plan application at various points

Calling plans are applied to route calls through the network at three points:

- *Ingress* —at the call's point of entry into the network, also called "source"
- *Egress* —at the call's point of exit from the network, also called "destination"
- *Transit* —applied to all calls traversing the network, regardless of their points of ingress and egress, if there is a DNIS match.

A calling plan can include all three call route application types.

### **Calling plan operation at ingress/source**

When a calling plan is applied at the ingress (or *source*) endpoint and the iServer receives a call setup request, the iServer follows the steps described below.

*Note: The iServer considers ALL configured routes only at Step 1 of the process. On executing each step after that, it obtains and learns a set of qualifying routes for the call, and executes the next step on this subset of all routes. Thus, with each step it removes routes that do not qualify, obtaining a smaller set of routes with each subsequent step.*

1. It looks up the originating gateway/endpoint, by its IP address or another attribute, such as subnet, or any of the protocol assigned names for the gateway.
2. If it finds the endpoint, it invokes the endpoint's device binding to fetch the appropriate calling plan. At this point, it also checks the endpoint's load factor (see *Setting session limits on calls, by endpoint* on page 79) to allow or disallow the call. It only considers this endpoint for routing the call if the call falls within the maximum call limit set on the endpoint.
3. If it selects this endpoint port for routing the call, the iServer retrieves the calling plan bindings and their respective binding priorities, for the calling plan and its routes.
4. The iServer looks at the called (destination) number and matches all available routes with this number to find the longest match. For instance, if the destination number specified in route A is 453 and that specified in route B is 4536, and the called number of the current call is 453611, the iServer picks Route B (longest match) over Route A.
5. When it has narrowed down and picked a set of routes the iServer applies the aggregate binding attributes of the routes to the call itself. For instance, if the time of day criteria does not match, then it bypasses the route for a more favorable route.
6. Once it finds a favorable route to route the call, the iServer executes the action specified by the route, which could be:
  - Rejecting the call
  - Transforming the called number during call setup

Hence, the iServer uses the following order of precedence for ingress call routing:

1. Load factor on endpoint/gateway
2. Calling plan binding priority (that is, the priority for the route, within the plan)
3. Longest match
4. Time of day

### **Ingress call routing application scenarios**

You can use ingress call routing for:

- Source tagging
- Normalized number routing

#### ***Source tagging***

You can tag an outgoing call at source such that a prefix is added to the destination number as it is dialed out.

This application is especially useful when using tech prefixes that must be prepended to all calls from a gateway. (A tech prefix is an identifying tag used by a gatekeeper for Cisco gateway selection within the zone or domain it is controlling.) For compatibility with Cisco gatekeepers and gateways, the iServer can use the tech prefix that is configured as part of a calling plan, to determine the Cisco gateway that it must route a call to.

For instance, if you wish all calls tagged with the number 080# to be routed through a certain gateway, you can configure a calling plan route as follows and assign it to the calling port:

Destination number/Destination length = \*/unspecified

New prefix = 080#

The route is applied at source

In this instance, the iServer tags all calls that originate from the endpoint this route/plan is assigned to, with the number 080#. When the iServer encounters a call with the destination number starting with 080#, it automatically uses the egress routes configured at the destination to route it to a gateway that has registered with it using the tech prefix 080#. See *Addition of prefix to destination endpoints* on page B-2 for the procedure to set this up using the CLI.

**Normalized number routing**

Most phone numbers in a Virtual Phone Network (VPN) allow for abbreviated dialing. Using ingress call routes, you can convert these abbreviated numbers to a normal format.

For instance, if you wish the extension number 4301 to be connected to the phone number 3014534301, you can configure a calling plan route as follows and assign it to the calling port:

Destination number/Destination length = 4301/4

New prefix = 301453

The route is applied at source

The resulting dialed number is 3014534301

In this instance, the iServer prepends the number 301453 to any call made by dialing the digits 4301, so the number actually dialed out is 3014534301.

**Calling plan operation at egress/destination**

In the case of egress call routing, the procedure is reversed from that at the point of ingress. Here, the iServer knows the called number, and must deduce the right gateway/endpoint to which to route it. When a calling plan is configured for egress call routing and assigned to an endpoint port, the iServer follows the steps given below to route a call.

**Note:** *The iServer considers ALL configured routes only at Step 1 of the process. On executing each step after that, it obtains and learns a set of qualifying routes for the call, and executes the next step on this select set of routes. Thus, it weeds out routes that do not qualify, obtaining a smaller set of routes with each subsequent step.*

1. It matches the destination number against the set of egress routes, sorted by route priority.
2. Once it identifies a set of routes by using the longest length match method, the iServer implements each route's action, which could either be a number translation or a rejection.
3. The iServer applies the time of day attribute and considers only those routes that qualify for the next step.
4. The aggregate binding for each route identifies the calling plan(s) that the route is part of. Since the iServer already has this knowledge, it now reads the device binding (and therefore the device priority) for each route.



5. The iServer selects the endpoint/gateway with the highest priority to route the call to. It now checks the load factor of the endpoint/gateway by considering the following two configurable policies:
  - Maximum calls that the endpoint can handle (see *Setting session limits on calls, by endpoint* on page 79)
  - Least-recently-used endpoint
6. If the endpoint/gateway's load factor permits it to accept the call, the iServer routes the call to it. If the load factor does not permit it to accept the call, the iServer considers the endpoint/gateway with the next highest device priority.
7. The iServer then selects the endpoint/gateway with the least utilization level. The utilization level is defined by the number of concurrent calls divided by the maximum number of calls.
8. If it finds a match, the iServer now checks the matched gateway/endpoint to see if it belongs in the same or "NULL" VPN group as the call originating gateway/endpoint. If it does not, then the iServer bypasses the gateway/endpoint, and repeats the selection process from step 4.
9. The iServer now checks the matched endpoint/gateway to see if it belongs to the same zone as the call originating gateway/endpoint. If it does not, then the iServer bypasses the gateway/endpoint and repeats the selection process from step 4.
10. The iServer then routes the call to the matched gateway/endpoint.
 

If neither route is a reject route (see "Reject routes" below), the iServer selects egress-leg call routes by filtering them in the order shown below. If there *is* a reject route on the plan, the iServer ignores the aggregate binding priority.

  - 10.1 Zone. A destination route for a different zone than that from which the call was placed is eliminated.
  - 10.2 Time of day. A route that is not active at the current time of day is immediately eliminated.
  - 10.3 Aggregate binding priority (i.e., route priority) within the calling plan
  - 10.4 Longest length of matched digits
  - 10.5 Device priority (also known as gateway priority)
  - 10.6 Load factor on endpoint/gateway

## 10.7 Utilization level on endpoint/gateway

## 10.8 VPN group

### **Reject routes**

A reject route is basically a routing rule that is an exception to a higher-level routing rule. Reject route status only applies to egress routes.

An example of reject routing:

A calling plan contains a positive route of 301, and a reject route of NPANXX=301501. The reject route, by being more specific, defines an exception to the positive rule, and therefore takes precedence, even if the positive route is of a higher priority. Therefore, any call to NPA 301 uses the positive route—*unless* it is to the 501 NXX within the 301 NPA, in which case the reject route takes over. Similarly, the more specific positive route 3015011 overrides that same reject route, even if the reject route has higher priority.

Egress routes can be defined as reject routes in RSM Console by selecting the Default Route checkbox in the **Modify Calling Plan Route** window's DNIS area.

Route bindings also have reject routing capability, but only when applied to egress routes.

### **Sticky routes**

An iServer call route can be tagged as *sticky*. This means that once that route is reached, the call either completes with that route, or the call setup is dropped.

*Sticky routing* forces the iServer's call hunting to cease at a point before reaching the end of a list of all routes that could potentially route a particular call. Once a sticky route is encountered, the call either completes with that route<sup>3</sup>, or the call is rejected. Note that the iServer's hunt logic always prevents hunting back to an endpoint that has already failed to complete the call.

The basic principles of sticky routing are:

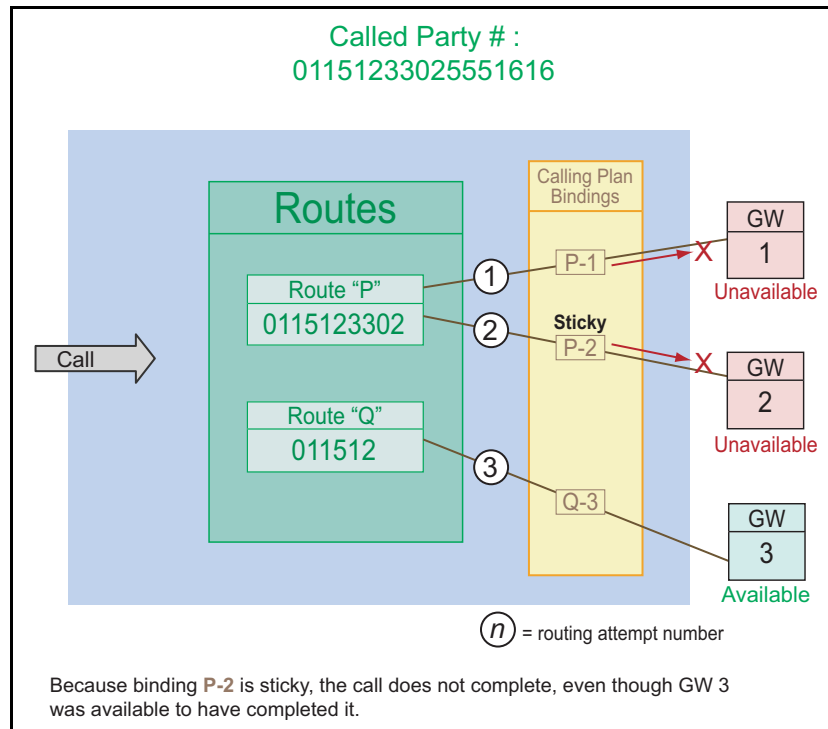
- Sticky routing applies to individual call routes.
- Because sticky routing controls call hunting, it applies only to egress routes.
- The *sticky* parameter is set either from RSM Console's *Modify Call Plan Route* panel, or using the `cli cr edit route name sticky [ enable | disable ]` command.
- The `cli cr list` command shows the sticky property for each route.

---

3. On any endpoint bound to the plan, provided it is of equal or numerically-lower binding or endpoint priority compared to the endpoint that didn't complete.

The steps below provide a simple example of the iServer's sticky routing feature. This example is illustrated in Figure 83.

**Figure 83. Sticky Routing Example**



1. A call comes into a network controlled by an iServer. For purposes of this example, we don't care how/where it entered the network.
2. On that network are three terminating gateways, G1, G2 and G3, all of which can provide suitable network egress.
3. Two egress routes, P and Q, can potentially route the call to an egress gateway. Route P is chosen first based on the rules of route selection (see *Calling plan operation at egress/destination* on page 480, step 10).
4. Route P is bound to gateways G1 and G2.
5. Route Q is bound to gateway G3.
6. Gateways G1 and G2 are cannot route the call. (They may be unregistered, down, max'ed out, or whatever. The reason does not matter.)
7. Gateway G3 is available to route the call.
8. Route P attempts to send the call to G1 (routing attempt 1).

9. Route *P* then attempts to send the call to G2 (routing attempt 2).
10. If *P*'s sticky route property is turned on, hunting stops with the last endpoint bound to *P* (G2), and route *Q* is not even tried. The call is rejected.
11. If *P* is not a sticky route, the iServer hunts to route *Q*, and the call completes through G3 (routing attempt 3).

**Note:** *For the configured sticky property to become active, the routing must be configured so as to lead to the call using that route, even though an endpoint bound to it may not present available resources for routing the call.*

### **Egress call routing application scenarios**

You can use egress call routing for:

- Time-based applications (including time of day, day of week, month, and year.)
- Route priority applications

#### ***Time of day example***

You can set different times of a day when a route should be applied at different endpoints. For instance if you wish all calls to the number 2404538322 to be routed to gateway A from 9:00 am to 5:00 pm and to gateway B from 5:00 pm to 9:00 am, you can:

1. Configure a calling plan route as follows:

Destination number/Destination length = 2404538322/10

New prefix = 2404538322

The route is applied at destination

2. Assign it to calling plan A.
3. In its aggregate binding properties, set the time of day attribute as follows:
  - Start time = 9:00 am
  - End time = 5:00 pm
4. Assign calling plan A to gateway A.
5. Assign the same route to calling plan B.
6. Under calling plan B, in its aggregate binding properties set the time of day attribute as follows:
  - Start time = 5:00 pm
  - End time = 9:00 am

### 7. Assign calling plan B to gateway B.

In this instance, the iServer uses calling plan A (and therefore gateway A) to route calls to 2404538322 between 9:00 am and 5:00 pm. It also uses calling plan B (and therefore gateway B) to route calls to 2404538322 between 5:30 pm and 9:00 pm.

### ***Route priority example***

You can prioritize the same route differently on two gateways such that one gateway gets picked for call routing over the other, is continued to be used until it reaches its load limit.

For instance, if you wish all calls for 5122334509 to be routed to gateway A (with a maximum call limit of 700 calls) first before they are routed to gateway B (with a maximum call limit of 300 calls), you can:

#### 1. Create a call route as follows:

Destination number/Destination length = 5122334509/10

New prefix = 5122334509

The route is applied at destination

2. Assign it to calling plan A.
3. In its aggregate binding properties, set the route priority as 2.
4. Assign calling plan A to gateway A.
5. Assign the same route to calling plan B.
6. In its aggregate binding properties, set the route priority as 1.
7. Assign calling plan B to gateway B.

In this instance, the iServer uses calling plan A (and therefore gateway A) to route calls to 5122334509 until the maximum call limit of 700 calls is reached on gateway A. On encountering the 701st call, the iServer starts routing calls to calling plan B (and therefore gateway B). While doing this, if the number of concurrent calls on gateway A decreases below 700, the iServer switches the routing of calls back to gateway A, and so forth.

### ***Stripping numbers at destination, example***

You can set up a calling plan such that the first few digits of the dialed number get stripped before it reaches the destination. This application is useful when a destination endpoint is configured to receive dialed numbers in a specific format.

For instance, if you wish to make a call to Mexico City and the carrier in Mexico only accepts numbers without the long distance code attached, you can create a route as follows and assign it to a calling plan:

Destination number/Destination length = 5255/unspecified

New prefix number = *blank*

The route is applied at destination

In this instance, the iServer routes the call as follows:

1. It matches the first few digits of the dialed phone number with the destination number, and uses this route only if those digits are 5255 (i.e., the code for Mexico City).
2. Since the new prefix is left blank, it strips the first few digits of the called number, replaces it with a blank and dials out the number without the digits 5255.

### ***Transit routes***

Transit routes are also called *unbound* routes (or *global* routes). These routes are additional to the system's source and destination routes, and are applied to all calls with a matching DNIS (after application of the ingress route).

In summary, transit routes have the following characteristics:

- They are applied:
  - After the source calling plan has been applied, and
  - Before the destination calling plans are applied.
- They apply to *all* calls.
- They function like normal routes (i.e., there is a destination pattern/prefix, etc).
- They apply to DNIS only. In particular, they are applied to CDR field no. 31 (see Table 15 on page 396).
- They cannot be reject routes.
- They are applied irrespective of any streams configured on the system.

### **Practical Applications**

- Transit routes can reduce the overall number of routes in the system.
- Reject Transit routes can be powerful in blocking calls on a global level.

## Other calling plan application scenarios

You can use calling plans to configure the following scenarios:

- Blocking incoming calls to a gateway port
- Forwarding outgoing calls from a gateway port
- Filtering outgoing calls from a gateway port
- ANI switching

### **Blocking incoming calls to a gateway port**

You can configure a gateway port with a calling plan route such that specific incoming calls at that port are blocked and not received. You can use this feature if you wish only certain gateway ports to receive calls from a specific number or location. For instance, if you wish all incoming long distance calls from your Virginia office to be received at only one port, you can configure two calling plan routes as follows and assign them to all the remaining ports on your network.

#### **Route 1:**

Called Party #/Length = 703/10

new Prefix = 703

Applied at: Egress

#### **Route 2:**

The route is a reject route

Applied at: Egress

In this instance, the iServer routes the call as follows:

1. It matches the first three digits of the dialed phone number with the Called Party number, and uses this route only if those digits are 703 (i.e., an area code in Virginia). If these digits are not 703, it does not route the call through this gateway port, and continues its search of a matching plan route.
2. It replaces the matching digits (i.e., 703) with the new prefix (which is also 703), and routes the call as-is.
3. When the reject route is encountered, the call is rejected.

**Forwarding incoming calls at a gateway port**

You can configure a gateway port with a calling plan route such that all incoming calls at that port are forwarded or rolled over to a different number. For instance, if you wish all incoming calls to your office number (4388846, for example) to be forwarded to your cell phone number (5109534441, for instance), you could configure a calling plan route as follows and assign it to the gateway port:

Destination number/destination length = 4388846/7

New prefix = 5109534441

The route is applied at destination

*Note: In this application scenario, you must establish a forwarding route using RSM Console. Refer to RSM Console online help for information on creating such a route.*

In this instance, the iServer routes calls as follows:

1. It matches the destination number 4388846 with the incoming phone number and uses this route only if the phone number matches this number. If the numbers do not match, it continues its search for a matching calling plan route.
2. It replaces the matching digits (in this case, 4388846) with the new prefix (in this case, 5109534441) and dials out this new number, which is your cell phone number, 5109534441.

**Filtering outgoing calls from a port<sup>4</sup>**

You can configure a gateway port with a calling plan route so that only outgoing calls to certain numbers are routed out of that port. This feature is useful if you wish to allow only calls to certain numbers or locations (such as local calls) from a gateway port. For instance, if you wish to allow only calls made to any number in Rockville, Maryland (area code 240) from a certain port, you can configure two calling plan routes as follows and assign them to the port:

Destination number/Destination length = 240/10

New prefix = 240

The route is applied at source

4. The feature in this example is not supported in this release, but is expected to be supported in a future release. It is included here for completeness.



Here, the iServer routes any call as follows:

1. The iServer matches the destination number with the dialed phone number, and routes only calls starting with the numbers 240 (i.e., the area code for Rockville, Maryland) from this port. If the dialed number does not start with the digits 240, the iServer does not use this calling plan route for the call, and continues its search for a matching calling plan route.
2. The iServer replaces the matching digit sequence (i.e., 240) with the new prefix (which is also 240) and dials out the phone number as is.

*Note: In this particular example, checking the No Digit Strip option would have had the same net effect on the dialed number.*

You can also use this feature to prepend an area code or a specific sequence of digits to an outgoing call. For instance, if you wish all outgoing calls from a gateway port to be dialed out with a “9” as the first digit (which could be the number to dial out using the PSTN line), you can configure a calling plan route as follows and assign it to the port:

Destination number/Destination length = *blank/unspecified*

New prefix = 9

The call is applied at source

In this case, the iServer routes the call as follows:

1. Since the destination number is left blank and the destination length is unspecified, it routes any call of any length through this port.
2. Since the new prefix is 9, it prepends the digit 9 to the dialed phone number and routes out the call.

## Permissive dialing

In a telephone network environment, dialing plans or numbering schemes for countries/geographic areas/enterprises often change. In order to ensure a smooth transition, voice providers and carriers must be able to provide support for both the old and new plans before finally discontinuing the old.

The NexTone iServer supports permissive dialing, which allows voice providers and carriers to use calling plans or numbering schemes. When configured for calling plans, the iServer maintains both the old and new dialing plans/numbering schemes in its database. The voice provider/carrier can then create various routes that are bound to the calling plan.

In many instances area codes, city codes, or exchanges may change and the accustomed pattern of dialing for the user changes as well. Using calling plans and routes, the iServer can accept both the old and new string. In each of these routes, the provider/carrier can configure a part of, or the whole dialed number to be translated/converted/replaced with a different number, thus ensuring conformance to the new dialing plan/numbering scheme. The provider/carrier can then apply this calling plan at the source.

When the iServer encounters a dialed number that follows the old dialing plan/numbering scheme, it applies the appropriate call route specifications to the dialed number, and converts it to conform to the new dialing plan/numbering scheme. Thus, the resultant number that is actually dialed out of the iServer to the destination conforms to the new dialing plan/numbering scheme.

### ***Adding a route***

Most carriers utilize permissive dialing as a grace period during the migration process. In the case where a single area code changes, the calling plan change simply requires an additional route to be added, as shown in the following example.

- Old area code - **919**
- New area code - **336**

1. Enter the new route to be applied in an egress GW.

**Figure 84. Adding a New Route to an Egress Gateway**

**Add Calling Plan Route**

Add new route

Partition: **admin**

Route Name: PermDialEx

☐ ANI type ☒ DNIS type

**ANI**

Calling Party #: Length: ☐ Length unspecified

Prefix: Usage [help](#)

**DNIS**

Called Party #: 919 Length: ☐ ☒ Length unspecified

Prefix: 336 ☐ No Digit Strip

☐ Default Route

☐ Reject

☐ Sticky

**Properties**

Applied at: ☐ Ingress ☒ Egress ☐ Transit

Template: ☐

Add Clear Close

2. Enter the new area code in an existing route.

**Figure 85. Entering the New Area Code in an Existing Route**

**Add Calling Plan Route**

Add new route

Partition: **admin**

Route Name: PermDialEx

☐ ANI type ☒ DNIS type

**ANI**

Calling Party #: Length: ☐ Length unspecified

Prefix: Usage [help](#)

**DNIS**

Called Party #: 919 Length: ☐ ☒ Length unspecified

Prefix: 919 ☒ No Digit Strip

☐ Default Route

☐ Reject

☐ Sticky

**Properties**

Applied at: ☐ Ingress ☒ Egress ☐ Transit

Template: ☐

Add Clear Close

### ***Country-wide permissive dialing***

To implement permissive dialing for an entire country or on a large quantity scale, it would be necessary to import a text file into the iServer's database. The format of the text file can be found in the iServer Configuration Guide.

For assistance, contact the Technical Assistance Center at NexTone Communications, Inc.

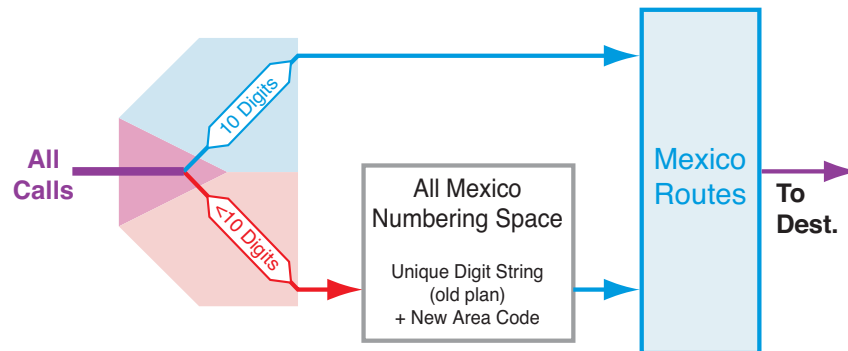
### **Example**

Here is an example where permissive dialing was temporarily required for an entire country.

Mexico changed from accepting 7-, 8-, or 10-digit dialing (depending on the city) to mandatory 10-digit dialing. For a period of time, the carriers accepted both the old and new plans. To support the migration, tens of thousands of new

routes were added to the iServer, to cover the two differing methods of dialing, as shown in Figure 86.

**Figure 86. Permissive Dialing, Mexico Example**



In this example, a call coming in with 10 dialed (DNIS) digits will be passed directly to the list of routes for Mexico. A call with fewer than 10 dialed digits will be passed through an additional step, where the correct new area code will be determined based on the old rules, and the new area code prepended to the number before

In the Mexico example, a text file can be created from which the Gen\_CP utility<sup>5</sup> can produce the appropriate routes. The input file would contain:

```
City, Area_Code, Start_Number, End_Number, New_Area_Code,
Calling_Plan, Priority [, Calling_Plan, Priority, ...]
```

Text file fields for this example are defined in Table 33.

5. See *Bulk route creation* on page 470.

**Table 33. Example Text File Field Definitions**

Field	Definition
Calling_Plan	Calling Plan name. Destination carrier/Gateway
City	City/State name
Area_code	Area code
Start_number	Start E.164 address in address range
End_Number	End E.164 address in address range
New_area_code	Change to Area_code if necessary
Priority	Priority of Calling Plan binding (Higher # = Higher Priority)

**Sample CSV file**

Below is an example of a CSV (comma-separated value) file of the type that could be used to implement changes requiring many new routes, as with the permissive dialing scenario just given:

```
GDL_Jalisco,01152,39730000,39739999,0115233,
H323_CARRIER_CP,1, SIP_CARRIER_CP,0
```

```
MTY_NuevoLeon,01152,87070000,87079999,0115281,SIP_CARRIER_C
P,0, ALT_H323_CARRIER_CP,0
```

...

NexTone provides a tool (known as Gen\_CP) by which such a CSV file can be read into the iServer, to avoid having to set up all this information item-by-item in RSM Console. See *Bulk route creation* on page 470 for details.

**Basic ANI manipulation**

With the ANI (calling party number) manipulation feature the iServer allows the modification or generation of the ANI on the egress call leg based on rules specified by the user.

To enable this feature, the following parameters must be specified:

- **Calling Party #** - the source pattern to look for to apply the rule. This pattern can be blank, which means it will match everything.
- **Length** - the source pattern length. If left unspecified (**Length unspecified** option left unchecked), the source pattern will be matched as a prefix against the ANI, which can be any length. If the source length is specified, the ANI must be exactly **Length** digits long.
- **Prefix** - New Prefix. Generation criteria is also specified here.

*For example, Route1:*

Src = 9111

SrcLen = 11

SrcPrefix = 0119111

Input = 91114546342 → Route1 → 01191114546342

The Generation symbols are as follows:

? = Generate a single random digit in the range [0-9].

%[xy] = Generate a single random digit in the range [x-y].

*For example, Route2:*

Src = 9111

SrcLen = 11

SrcPrefix = 011911145%48????\$

Input = 91114546342 → Route2 → 01191114556473

The “\$” at the end of the prefix indicates not to suffix the remaining unmatched digits in the input, as in the example with Route1, where 4546342 are attached to the end of the new ANI by default. In this example, if the SrcPrefix was 011911145%48???, the new ANI would be 011911145564734546342.

### **Applying ANI manipulation plans to calls**

ANI manipulation plans are configured as routes, and added to calling plans. These calling plans may be configured on the source endpoint (ingress leg) of the call or the destination gateway (egress leg) of the call. Applying ANI manipulation plans at the source endpoint is a generic application at the source for all calls emanating from the source that match the route. The selection of a destination gateway is not made based on the ANI routes or the degree of match. In general, first the destination gateway is selected based on call routing criteria (specified elsewhere) then the ANI routes of the destination gateway

are applied. The best ANI route match is one that gives a longest match in terms of the pattern.

## **File-based ANI manipulation**

A list of ANIs can be placed into an editable text file, and subsequently used to provide means of sending ANIs from the list as calls come through that route. This can be useful when a customer wants it to appear as if many calls entering the network from the same source are actually coming from different sources, but wants the ANIs sent to the vendor to be real numbers, supplied from a list, rather than random numbers (which can be achieved without this feature by simply placing wildcard characters in the “ANI Prefix” field).

### ***Feature setup overview***

Setting this feature up involves three basic steps:

1. Set up a plain text file containing the ANIs to be sent.
2. Create an egress ANI route with a ANI prefix that names the text file created in step 1.
3. Bind the egress ANI route to an existing egress calling plan.

### ***Text file requirements***

The ANI list text file must meet the following requirements:

- It must not be empty.
- It must not contain any blank lines.
- The numbers in it must be of the required ANI format, because the iServer does not validate them.
- Normally it is placed in the `/usr/local/nextone/bin` directory. It can be put into any directory you choose, but putting it somewhere else, requires specifying the complete path to the file, instead of just its name, when setting up the ANI egress route that uses it.

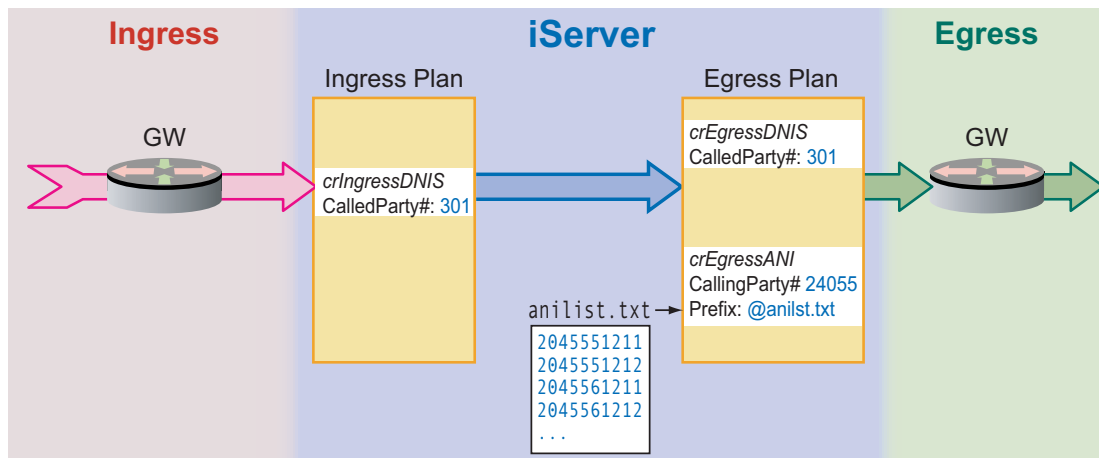
### ***Feature example***

This simple example illustrates one use to which this feature might be put.



Assume we have a simple database in which there is are two calling plans (one ingress, one egress), and three routes (one DNIS ingress, one DNIS egress, and one ANI egress). Figure 87 illustrates this.

**Figure 87. File-Based ANI Manipulation, Example**



1. A call is placed from ANI 204-556-1213, to DNIS 301-222-3333.
2. The call enters the iServer via the bound ingress plan's DNIS ingress route (*crIngressDNIS*), which prepares selected call parameters for egress.
3. In the egress plan, DNIS egress route *crEgressDNIS* applies, providing egress on the bound gateway.
4. In the egress plan, *crEgressANI* applies. This ANI egress route takes one entry from the *anilst.txt* file (the file the ANI Prefix route entry points to), forces that entry into the ANI field in the outgoing call setup, and increments the index into the file.

Another call from the same source results in the next entry in the *anilst.txt* file, and so on, until the bottom of the file is reached. The index into the file is then reset, and the process begins again.

### **Detailed setup procedure**

Once the text file is created, the remaining steps in setting up this feature can be performed via either RSM Console or CLI commands, as given below.

**RSM Console procedure**

Begin by creating the ANI egress route:

1. Open RSM Console's **Calling Plan** utility (double-click the iServer in the main map window, and choose Utilities → Calling Plans).
2. Click the Add button under the list of defined routes.
3. Enter a name for the new route, and confirm that the ANI radio button is selected.
4. Enter route selection parameters for Calling Party #, Length (or Length unspecified), as desired.
5. In the Prefix box, enter the full name of the file, preceded by an "at" sign, as shown in Figure 88 (in the example, @V1ANImanip.txt). Note that if the file was not placed in the /usr/local/nextone/bin directory, you must specify the complete path to the file, not just its name.

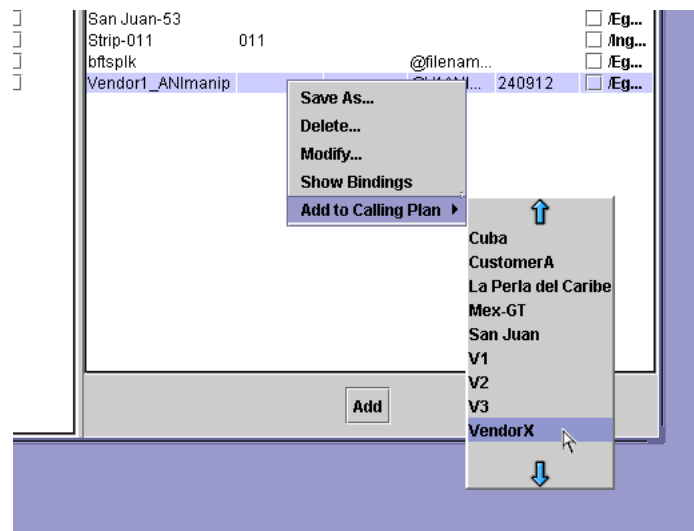
**Figure 88. Specifying an ANI Input File**

6. In the Properties frame, click Egress.
7. Click Add, then click the Close.

Next, bind the route to an existing egress calling plan:

1. Locate the route you just added in the list of routes shown in the rightmost frame of the **Calling Plan** utility window. (If the list is too long to fit in the window's height, you may need to scroll to the bottom to find it.)
2. Right click on the route's name. Move the pointer over Add to Calling Plan >, then click on the name of the plan from the list that opens. (See Figure 89).

**Figure 89. Binding the ANI Egress Route to the Egress Calling Plan**



3. Close the **Calling Plan** utility when finished.

### **CLI procedure**

This procedure is provided in case RSM Console is unavailable, or you only have an RSM Console version prior to 3.1, but the RSM Console procedure is preferred. Confirm that the egress calling plan to which the new route will be bound already exists. If not, create that calling plan first, *then* perform this procedure.

**Note:** *If you have a version of RSM Console prior to 3.1, and set up the ANI egress route for this feature via this command-line procedure, DO NOT edit the resulting route through RSM Console. Doing so will remove the ANI prefix entry that points to the file.*

Begin by creating the ANI egress route.

1. Log onto the iServer.

2. Enter the command:

```
cli cr add cr-name calltype dest srcprefix @file spec
```

Where:

- cr-name is the name of the new call route you are creating, and
- file spec is either the file's name, or the complete path to it, if the file was not placed in the /usr/local/nextone/bin directory.

3. Now, to bind the new route to its calling plan, enter this command:

```
cli cp add cp-name cr-name
```

Where:

- cp-name is the name of the calling plan to which you are binding the new route, and
- cr-name is the name of the ANI egress route you created in step 2.

### Feature operation notes

- When the egress calling plan is selected, the original ANI is replaced by one selected from the file named in the ANI Prefix field of the egress ANI route. Each time the route is selected, the index into the file is incremented (and therefore, the next ANI number in the file is used), until the bottom of the list is reached. After the last number in the file is sent, the index resets, and the first number in the file is sent again, etc. Note that if the iServer is reset, the index will begin at the top of the file again, regardless of which entry was last sent.
- If an ANI file associated with a call plan binding is edited or replaced with a file of the same name while the iServer is running, the ANI selection index rewinds to the start of the file.
- If an ANI file associated with a call plan binding is not found or is empty, the original ANI is not replaced.
- **Upgrades:** The iServer upgrade procedure does not back up any user-created ANI files, nor does it copy them over to the new version. Such files must be manually copied following the upgrade.
- **SIP:** If the egress endpoint is a SIP gateway, the name in the *From* and *Contact* headers is replaced in the outgoing INVITE.
- **Hunting:** With this feature, the ANI will be replaced for every hunt attempt of a call, so each *placed* call will not necessarily contain the very next ANI entry in the list.

- To remove this feature, delete the aggregate binding (i.e., the binding between the egress ANI route that contains the filename, and the calling plan), or delete the egress ANI route.

## DNIS default route

The iServer only treats DNIS transformations as valid routes, even if no DNIS transformation is required (such as those routes that only do ANI switching for any DNIS), by using the “Default Route” flag. DNIS default route is the ability to create a route with an empty dialed number and cause it to always match. An example configuration is shown in Figure 90.

**Figure 90. DNIS Default Route**

The screenshot shows the 'Add Calling Plan Route' dialog box. The 'Add new route' section has a 'Partition' dropdown set to 'admin' and a 'Route Name' text box containing 'DefaultEx'. Below this, the 'ANI type' radio button is unselected, and the 'DNIS type' radio button is selected. The 'ANI' section contains fields for 'Calling Party #', 'Length', and 'Prefix', with a 'Length unspecified' checkbox. The 'DNIS' section contains fields for 'Called Party #', 'Length', and 'Prefix', with a 'Length unspecified' checkbox checked and a 'No Digit Strip' checkbox. The 'Default Route' checkbox is checked and highlighted with a grey oval. Below it are 'Reject' and 'Sticky' checkboxes. The 'Properties' section has 'Applied at' radio buttons for 'Ingress' (selected), 'Egress', and 'Transit', and a 'Template' checkbox. At the bottom are 'Add', 'Clear', and 'Close' buttons.

## Source port selection

iServer provides the ability to select a source port based on a call's ANI and DNIS information. This functionality works as described in this section.

### **Source-side uports**

Source-side uports serve the following purposes:

- Identification of the customer in a logical fashion
- Channeling the call to an egress gateway in a manner involving the fewest routes (simplifying configuration) through the use of tags and/or streams
- Perform digit manipulation at the call's point of ingress into the network

### **Port selection algorithm**

The iServer source port selection algorithm is structured as follows:

1. Find src regid based on one of the following IP addresses:

- Src IP address from Setup
- Src Signaling Address in Setup
- Contact in INVITE
- Via in INVITE

If source regid is found, the uport selected as part of subsequent steps *must* be a uport within this regid or the step is skipped. If no source regid is found, any regid/uport found in one of the following steps is used.

2. If a subnet entry containing any of the following exists, it is used:
  - Src IP address from Setup
  - Src Signaling Address in Setup
  - Contact in INVITE
  - Via in INVITE exists, it is used.
3. If a trunk group is available in the signaling message and matches one on the iServer, it is used.
4. If an entry is provisioned on the iServer with the ANI specified in the signaling message, it is used.
5. If an H.323 id exists in the signaling message, and matches one in the iServer database, it is used.
6. Among all ANI- and DNIS-based source routes which match the call, the one with the longest match is used to determine the uport. If a regid was determined in step 1, only the routes for the uports on that endpoint are examined.

**Example**

A gateway (regID *att*) with two ports (*0* and *1*) is configured as follows:

- att/0 has two routes configured:
  - ANI route 555/301555, and
  - DNIS route 33/01133.
- att/1 also has two routes of its own:
  - ANI route 5556/3016666, and
  - DNIS route 331/01131.

If a call comes from *att*, with ANI 5556001, DNIS 3311001, the iServer will select uport 1 (because of the longer ANI match) and the DNIS will be translated to 0113311001.

**iServer trunk group support**

A *trunk* is a logical grouping of call circuits that connect network nodes (such as switches). Each trunk group bears an identifier, known as a *trunk group ID*. Trunk group ID's can be used to identify the source of the call, such as a peering partner or transport user-customer.

Each endpoint can be configured with trunk group parameters that identify the source gateway of an incoming call. The iServer can relay this information to the egress leg of the call or it can suppress or replace it with a different string (up to 24 characters). Trunk group IDs are typically supported by Cisco gateways running H.323 v4 protocol (IOS 12.2(11)T).

The basic purpose behind trunk grouping is to give the device requesting a call set up the ability to tell the session controller which egress path it wants used.

Trunk Groups are supported for both H.323 and SIP, though the two protocols use different parameters to communicate trunk group information. A trunk group ID is an alphanumeric string with a maximum length of 24 characters, which can contain white space. This information is written into the CDRs.

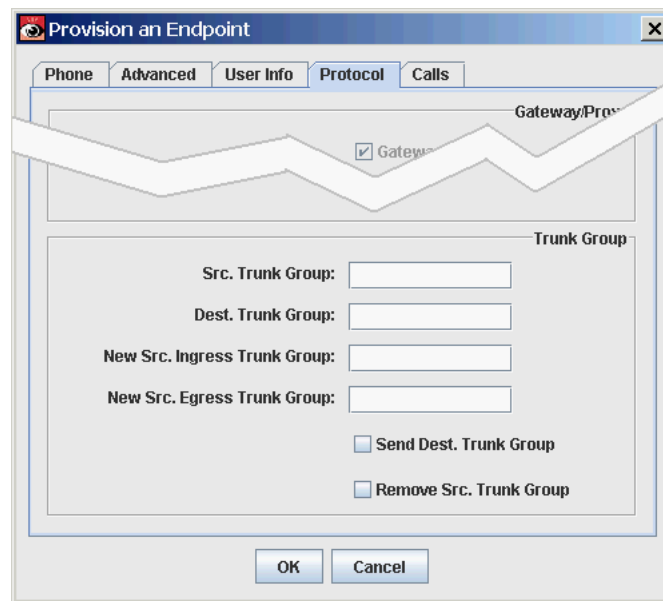
**Call setup source fields**

When an H.323 call setup request arrives at the ingress gateway, the setup message can contain two fields, *sourceCircuitID* and *destinationCircuitID*. A SIP INVITE has fields corresponding to these, which differ based on whether it is a PointOne or Nortel gateway (see "SIP Services" on page 194 for details). These fields map to the iServer's *Source Ingress Trunk Group* and *Source*

*Egress Trunk Group* parameters. These received fields can be read and passed through by the iServer. They can also be altered or removed before the egress setup request is sent to the egress gateway. The iServer can also supply values for blank *Source Ingress Trunk Group* and *Source Egress Trunk Group* fields.

Figure 91 shows the trunk group fields found on the *Modify (endpoint)* dialog's *Protocol* tab.

**Figure 91. Trunk Group Support in RSM Console**



### ***Trunk group parameter descriptions***

The parameters shown in Figure 91 provide the following capabilities:

- **Src. Trunk Group**  
Calls with setups having a `sourceCircuitID` matching this string can be routed through this endpoint/port.
- **Dest. Trunk Group**  
This parameter can be used two ways: 1) for destination port selection, and 2) to insert destination trunk information into the egress leg of the call, if none is present in the ingress setup message.
- **New Src. Ingress Trunk Group**  
This parameter overrides or supplies missing source trunk information from the incoming setup. The supplied information is passed on to the egress leg setup, if the **Remove Src. Trunk Group** option is *not* selected.



- New Src. Egress Trunk Group

This parameter can override or supply missing incoming leg destination trunk group information. *If* the Send Dest. Trunk Group option is checked, the supplied information is passed on to the egress leg call setup.

- Send Dest. Trunk Group

Controls the insertion of destination trunk information into the egress leg's setup message. Selecting this checkbox tells the iServer to populate the egress leg setup's *destination trunk group* field with either:

- the received destination trunk group field (if one is supplied by the incoming call setup request), or
- the contents of the Dest. Trunk Group field (if one is supplied on this dialog panel).

- Remove Src. Trunk Group

Checking this option removes destinationCircuitID from the egress leg setup request, if one was supplied in the ingress leg setup request.

*Note: The specification of trunks in H.323 messages conforms to the H.225 protocol standard.*

### **Trunk group data pass-through options**

For clarity, Table 34 provides a list of egress leg setup message trunk group field contents, based on the selected pass-through options. The cases assume both source and destination circuit ID fields are populated in the incoming call setup. Blank fields in ingress setups will be blank in the egress setup unless that field is supplied on the **Modify [endpoint]** panel.

**Table 34. Trunk Group Circuit ID Data Pass-Through**

Selected Options		Egress Leg	
Send Dest	Remove Src	destinationCircuitID	sourceCircuitID
unchecked	unchecked	(none)	ingress leg src
checked	unchecked	ingress leg dest	ingress leg src
checked	checked	ingress leg dest	(none)
unchecked	checked	(none)	(none)

**Trunk group implementation and configuration**

iServer's trunk group implementation allows VoIP carriers to bind iServer ports (a.k.a. uports) to specific Trunk Groups on the source gateway. Figure 92 below provides an example in which the gateway "MY\_GATEWAY" has 3 u-ports, with trunk groups CARRIER\_A, CARRIER\_B and CARRIER\_C assigned to each, respectively. Note that each u-port has a different source calling plan assigned to it.

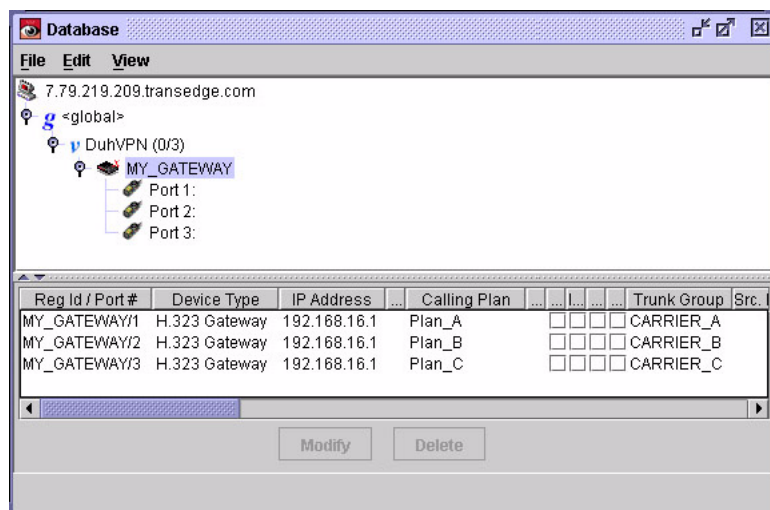
**Figure 92. Trunk Group U-Port Example**

Figure 93 shows a sample configuration of the source gateway called “MY\_GATEWAY” (non-essential output omitted), based on Cisco IOS 12.2.(11)T1:

```
MY_GATEWAY#sho run
Building configuration...
!
[ output suppressed ]
!
hostname MY_GATEWAY
!
trunk group CARRIER_A
!
trunk group CARRIER_B
!
trunk group CARRIER_C
!
[ output suppressed ]
!
voice-port 1/0/0
    trunk-group CARRIER_A
!
voice-port 1/0/1
    trunk-group CARRIER_B
!
voice-port 1/1/0
    trunk-group CARRIER_C
!
voice-port 1/1/1
```

**Figure 93. Source Gateway, Sample Configuration**

In this example, if a call originates from voice-port 1/1/0<sup>6</sup>, its the ingress trunk group field is automatically tagged by the gateway as CARRIER\_C. When the call reaches the iServer (via the ingress leg), this field is matched against the configured source trunk groups on all the u-ports of this gateway. In this particular example, the third u-port will be selected, because it is configured to accept calls from the trunk group CARRIER\_C. Consequently, the source calling plan “Plan\_C” (and any other policies such as maximum calls) will be applied. Moreover, the ingress trunk group name is always relayed to the egress leg, whether or not there is a matching trunk group configured on the iServer.

---

6. “1/1/0” is just the router’s notation for a voice port.

The iServer associates the ingress call to the first u-port of the endpoint if there is no matching trunk group configured on the endpoint. The only other case where a specific u-port, other than the first, will be selected is when an H.323 ID or E.164 is sent in the Source Alias field of a Setup message and matches any of the u-port's H.323 IDs or Extensions.

For more information on how to configure gateways to route calls based on trunk group, refer to the gateway vendor's documentation.

### ***Practical applications***

When placed as a border element in a VoIP network, the iServer enables the provider to insert trunk group identifiers into incoming call setups. This enables the provider to select ports based on some criteria (like H.323 id, trunk group, IP address, etc), and either supply a new trunk group (if it is missing) or override the incoming one with a new one. The source trunk group of the outgoing setup in this scenario would depict the source. The destination gateway may in turn use this information to execute source-based policy.

If an iServer gets an ingress setup message with the destination trunk group set, and the trunk group is found to exist on an endpoint defined in the local database, the iServer skips its routing logic, and routes the call to that endpoint.

This technique is useful in relaying source information from a border element to the core network and then relaying destination information from the core to an edge gateway. The destination trunk group setting in an egress setup message can enable many applications where the iServer sits in the core of the network with all the routing information and the iServers have the ACLs (access control lists) defined for endpoint access.

A typical application of trunks is where the routing information is configured "outside" the iServer. This is usually the scenario where a CSC/ESC layout is employed. The routes are provisioned at the CSC (usually an iServer, but can be replaced by any routing entity which can supply destination trunk information to the ESC) and the trunks are provisioned on the ESC.

Another configuration where this may be used is where the majority of calls on the iServer are "switched" rather than routed. In this case, the selection of the destination gateway is made at "ingress". Note that applying the selection criteria at "ingress" means that the call is essentially routed before any aggregation criteria like destination calling plans or transit routes are used. This means that an iServer configured with trunks can essentially be used to "route" calls without any routes (and hence we describe this by the word switching rather than routing).

**Trunk-based routing examples**

Here are examples where trunk group routing can be useful.

**Normal iServer routing:**

- Source Regid selection (IP address)
  - Source Port Selection (calling plans/routes/trunk/h.323id, etc)
  - Transit Routes
  - Destination Calling plan and routes
  - Destination selection

**Trunk Group switching:**

- Source Regid selection
  - Source Port Selection (calling plans/routes/trunk/h.323id, etc)
  - Destination selection

**Scenario 1: Call comes into the iServer with destination trunk set to X**

In this case, the iServer skips it's routing table lookups and searches for an available destination in the X trunk group. The iServer may use static or dynamic hunting to walk through the list of potential destinations servicing X.

**Scenario 2: Call comes into iServer with source trunk group set to X**

In this case, the iServer uses X to select a source port in the database.

**Scenario 3: Call comes into iServer, where the src port it comes in on supplies the destination trunk (Dest. Trunk Group field) as Y**

In this case, the iServer supplies (or overrides, if there is a destination trunk group in the INVITE) the destination trunk group with Y and searches for available destinations in trunk group Y.

## Emergency Call Admission Control

When an endpoint or iServer system reaches its configured limit (whether in number of calls, or bandwidth), no more calls can be placed through it. However, government regulations mandate that emergency calls *must* be placed. This chapter describes how you can configure the system to do so, while still protecting against fraud and attack.

### Overview of emergency CAC procedures

The steps in setting up emergency call admission control include:

1. Establish which numbers you want to reserve for emergency calling, and set up the list (see *Emergency number provisioning* on page 512).
2. Determine the level of control at which you want to provision each emergency number:
  - Emergency numbers are provisioned at the realm or global level
  - CAC limits are provisioned at the endpoint, iEdge group, or global level.
3. Set up call limits for the levels you need (see *Procedure: Implementing emergency CAC* on page 513).
4. Maintain the list of emergency numbers using the commands shown in *Available subcommands* on page 512 and the sections that follow it.

### Emergency CAC

The Emergency Call Admission Control (CAC) feature allows designating specific numbers as emergency numbers, and applying special CAC limits to calls to those numbers. Prepaid calling is permitted even when the limit has been exceeded.

Emergency numbers are configurable at the realm and global levels. If the dialed number in an incoming call request matches one of the configured emergency numbers (number only for global numbers, number and realm match for per-realm numbers), any emergency call CAC parameters that have

been configured are applicable to the call. A number may be defined so as to appear in multiple realms.

Emergency call CAC parameters consist of global, endpoint (that is, uport), and iEdge group (policy) parameters, correspond to the same non-emergency call CAC parameters and consist of both signaling (call) and bandwidth limits, as described later in this chapter. In each case, the traditional CAC limits must be exhausted before the emergency call CAC pools are used. Therefore emergency calls are allocated first out of normal CAC resources. When normal CAC resources are exhausted, non-emergency calls will fail, but emergency calls will succeed until emergency call resources are exhausted as well.

## Emergency calling limits

Limits pertaining to emergency calling apply at the endpoint, iEdge group (igrp), subnet, and global levels. They can be set to limit the number of concurrent emergency calls (global, igrp and endpoint), and bandwidth consumption (igrp only).

### Endpoint limits

Parameters controlling endpoint emergency calling limits include:

- x911calls (total calls)
- x911incalls (incoming calls)
- x911outcalls (outgoing calls)

These limits are set using the CLI's `iedge edit` command. For example:

```
cli iedge edit regid uport x911incalls incoming limit
```

### iEdge group limits

iEdge groups allow CAC limits to be applied to multiple endpoints, as described in *U-port group limits* on page 80.

Parameters controlling iEdge group emergency calling limits include:

- max911callsin (incoming calls)
- max911callsout (outgoing calls)
- max911callstotal (total calls)
- max911bwin (incoming call bandwidth)
- max911bwout (outgoing call bandwidth)

- `max911bwtotal` (total call bandwidth)

These limits are set using the CLI's `igrp` command. See *Administering iEdge groups with CLI commands* on page 82.

### **Global limits**

Global CAC limits on concurrent emergency calls can be set via `nxconfig.pl` (see *Global configuration using nxconfig.pl* on page 18) to provision additional emergency calling vports over the regular vport limit. Global parameters include:

- `max911vports` (maximum emergency call signaling ports; default value: 0)
- `max911mrports` (maximum emergency call media routing ports; default value: 0)

### **Capacity consideration when setting limits**

When setting limits for emergency calling, bear in mind that the scenario for which emergency call limits apply is when non-emergency calling limits have been hit (without regard to calls being normal or emergency), and would prevent additional call setups.

One implication of this is that if your normal (non-emergency) settings are configured so that the limit is at the point when your network runs out of bandwidth, emergency calls may push your network to the point that quality of service on existing calls drops to below what is acceptable. For this reason, it's best to keep your emergency calling limits to a number that is only a small percentage of your total calls (or bandwidth for igrps).

## **Emergency number provisioning**

A list of emergency numbers is created and maintained on an iServer by using the `cli` interface's `emergnum` command.

### **Realm-level provisioning**

Note that emergency numbers can optionally be limited to apply only within one realm. Not specifying a realm for a number makes it a *global* emergency number.

### **Available subcommands**

The `emergnum` command supports the subcommands:



- `list` (show currently defined emergency numbers)
- `add` (add a new emergency number)
- `delete` (delete an existing emergency number)

### Using the `list` subcommand

To display a list of emergency numbers that are already defined on the iServer, enter:

```
cli emergnum list
```

The resulting output tells how many numbers are defined, and shows the numbers (and defined realms, if applicable).

### Using the `add` subcommand

To add a new number to the list of numbers, use the `add` subcommand:

```
cli emergnum add number [realm]
```

If you specify a realm, the number is only defined as an emergency number if the call setup request to that number comes in from that realm.

### Using the `delete` subcommand

To remove a number that is already defined in the emergency number list, or to change a number from being realm-specific to being global, use the `delete` subcommand:

```
cli emergnum delete number [realm]
```

- If you specify a realm, and the number is currently defined in *more than one* realm, the number is removed only from the realm you name, and remains specific to any other realms for which it is defined.
- If you specify a realm, and the number you specify is currently defined in *only one* realm, the number remains in the list, but becomes a global emergency number.
- If you do not specify a realm:
  - If the number is defined globally, it is deleted.
  - If the number has realms defined for it, the command has no effect.

## Procedure: Implementing emergency CAC

Use these general steps for setting up emergency call admission control:

1. Log onto the iServer so that you can access the `nxconfig.pl` utility and CLI.

2. Obtain from NexTone and install a license file that enables emergency CAC on your iServer. For information on getting and installing license files, see *Obtaining an MSX license* on page 101 and *Installing an MSX license* on page 102.

3. Set the global quantity of emergency ports using the commands:

```
nxconfig.pl -e max911vport -v number of signaling ports
nxconfig.pl -e max911mrports -v number of media-routed ports
```

4. Define additional limits for a specific endpoint, if desired, with:

```
cli iedge edit regid uport [x911calls | x911out | x911in] number of calls
```

5. Define additional limits for groups of endpoints (iEdge groups), if desired, with:

```
cli igrp edit igrp name [max911callstotal | max911callsout |
max911callsin | max911bw | max911bwin |
max911bwout] limit
```

If you need to create a new igrp, or want other information on how igrps work, see *U-port group limits* on page 80.

6. Enter your list of designated emergency numbers into the iServer, using the add option of the `emergnum` command (described in *Emergency number provisioning* on page 512):

```
cli emergnum add emergency number [realm]
```

## Related CDR field

CDR field number 83, named `e911-call`, contains a flag indicating whether a call was classed as an emergency call, based on being defined as an emergency number. If the call was to a defined emergency number, this field contains `e911`. Otherwise, it is blank.

## Authentication and authorization of emergency calls

Emergency calls are subject to the same authentication and authorization controls as non-emergency calls. See Chapter 26, , on page 364 for more information on the iServer's authentication and authorization services.

## Support for interworking emergency calls

The iServer supports emergency CAC only for SIP-to-SIP calls.

## Configuring emergency calling with RSM

As an alternative to CLI commands, you can configure emergency calling using RSM Console or RSM Lite. Once you have installed a license file that enables emergency CAC on your iServer, configuration options specific to emergency calling become available. The following sections describe configuration procedures using RSM Console that parallel the command line procedures described earlier in this chapter.

### Accessing RSM Console

To begin any of these procedures, first start RSM Console and double-click the icon corresponding to the iServer on which emergency CAC is licensed. This opens the iServer window for that iServer.

### Adding emergency numbers

To add an emergency number to the iServer:

1. In the **iServer** window, select **Emergency Numbers** from the **Utilities** menu to open the **Emergency Numbers** panel. This panel lists any emergency numbers already defined for the iServer.
2. With the **Emergency Numbers** panel open, select **Add** from the **Edit** menu to open the **Add Emergency Number** window.
3. From the **Partition** list, select the partition to which this number should belong.
4. From the **Realm** list, select a realm if you want to restrict the number to a specific realm. If no realm is specified, the number will be treated as global.
5. Specify the emergency number in the **Phone Number** field.
6. Click **Add** to add the number.

After you add an emergency number it appears in the list found in the **Emergency Numbers** panel.

### Editing emergency numbers

To change a number's configuration:

1. Double-click on the number in the list in the **Emergency Numbers** panel. This opens the **Modify Emergency Number** window.
2. After making changes, click **Modify** to apply them and close the window.

**Deleting emergency numbers**

To delete an emergency number:

1. Click to highlight the number in the list in the **Emergency Numbers** panel.
2. Select Delete from the Edit menu.

***Configuring emergency calling limits at the global level***

You must specify the number of signaling and media routing ports you want to provision for emergency calling. These global-level values specify the number of additional ports to provision for emergency calling after the regular CAC limits are reached. These limits are set in your iServer's configuration. To specify these values:

1. Select Configure on the iServer menu to open the **iServerConfiguration** window.
2. Click the System tab.
3. In Maximum VPorts, enter the number of additional signaling ports you want to provision for emergency calling when the regular vport limit is reached. The default value is 0.
4. In Maximum MRPorts, enter the number of additional media routing ports you want to provision for emergency calling when the regular mrport limit is reached. The default value is 0.
5. Click OK, then Close to apply the changes.

***Configuring emergency calling limits at the endpoint level***

For each specific endpoint, you have the option to specify limits on the number of concurrent emergency calls it can handle. If you do not want to specify limits at the endpoint level, leave the settings at their default values of 0. In emergency call configuration, a value of 0 means that no additional emergency calls are allowed beyond the regular CAC limits.

These limits are specified within the endpoint's configuration. You can define these limits when you add a new endpoint or you can modify an existing endpoint.

1. For either a new or existing endpoint, the first step is to access the endpoint's configuration from the iServer window. If you are adding a new endpoint, select Add, then Endpoint, from the Edit menu to open an

**Add Endpoint** window. For an existing endpoint, access its configuration by double-clicking on its name in the list of endpoints found in the lower pane of the iServer window.

2. Once you are within the endpoint configuration settings, click the **Calls** tab and locate the **E911** group at the bottom of the window. In the **E911** group box, specify the following call limits for the endpoint:
  - Enter the total number of additional emergency calls to allow above the regular CAC limit in **Maximum Total Calls**.
  - Enter the number of additional incoming emergency calls to allow above the regular inbound CAC limit in **Maximum Ingress Calls**.
  - Enter the number of additional outgoing emergency calls to allow above the regular outbound CAC limit in **Maximum Egress Calls**.
3. Click **OK** to apply the changes.

### ***Configuring emergency calling limits at the iEdge group level***

You have the option to specify calling limits that apply to multiple endpoints by specifying limits at the iEdge group level. These limits include both call limits and bandwidth limits. If you do not want to set limits at the iEdge level, leave these options blank.

These limits are specified within the iEdge group's configuration. You can define these limits when you add a new iEdge group or you can modify an existing iEdge group.

1. For either a new or existing iEdge group, the first step is to access the group's configuration from the iServer window. If you are adding a new group, first select **iEdge Group** from the **Utilities** menu to open the **iEdge Groups** panel. Then, select **Add** from the **Edit** menu. For an existing iEdge group, double-click the appropriate group name from the list.
2. Once you are within the iEdge group configuration settings, locate the **E911** group at the bottom of the window. In the **E911** group box, specify the following call limits for the iEdge group:
  - Enter the number of additional emergency calls to allow above the regular CAC limit in **Max Call Total Legs**.
  - Enter the number of additional incoming emergency calls to allow above the regular inbound CAC limit in **Max Call In Legs**.
  - Enter the number of additional outgoing emergency calls to allow above the regular outbound CAC limit in **Max Call Out Legs**.

- Enter the amount of additional incoming bandwidth to allow for incoming emergency calls in Max Bandwidth In.
  - Enter the amount of additional outgoing bandwidth to allow for outgoing emergency calls in Max Bandwidth Out.
  - Enter the total amount of additional bandwidth to allow in Max Bandwidth Total.
3. Click Add or OK to apply the changes.

## Dynamic Call Hunting

The iServer supports call hunting in cases where the preferred destination for a call is unavailable or drops the call due to a temporary failure condition at the destination<sup>1</sup>. This type of event triggers the iServer to establish the call with the next most preferred gateway based on the routes configured on the iServer.

A stateful iServer processes every end-to-end call as an ingress leg and an egress leg. Once the ingress leg is set up, the iServer initiates a hunt on the egress leg until that call leg has been successfully set up. Once the egress call leg has been successfully set up, the source and destination endpoints are connected, and the call proceeds.

Call hunting is either static or dynamic. Static call hunting is implicit to the system, as the criteria for it are:

- endpoint loading
- routing policies

### DCH criteria

The iServer performs dynamic call hunting (DCH) based on the following:

- On a failed egress call attempt, the next destination endpoint to be selected is one which is the best of the remaining set of endpoints which can route that call. The iServer will also hunt to the next egress endpoint if the attempted device isn't responding (for example, for a TCP timeout, or a Q.931 setup timeout).
- A maximum of 50 hunts or route advances can be set up on the system.
- Call hunting is enabled globally for the system or on a per-source-endpoint basis. The configuration on the source endpoint overrides the global system setting.
- Call hunting is dynamically disabled whenever the destination endpoint sends media in its Progress Message indication.

---

1. See "Cause Code Operations" on page 524 for information on conditions under which call hunting is initiated.

- Both SIP and H.323 calls are subject to hunting. Whenever the iServer forwards media in proceeding, hunting is disabled for that call.
- A call route or an endpoint may be designated as *sticky*. Once a sticky route or endpoint is contacted, the call either completes with that route or endpoint, or the call is rejected (see *Sticky routes* on page 482 and *Making an endpoint “sticky”* on page 78).

Every route hunt operation generates a CDR<sup>2</sup>. The CDRs generated for hunting are linked by a common call-id.

## Hunting triggers

Certain events in the system trigger the iServer to select the next most preferred gateway to establish the call. Hunt triggers are always based on the destination of the call. For H.323 to SIP calls, the triggers for SIP take effect. Following are some examples of call failure scenarios that trigger call hunting.

### **Termination entity is H.323 gateway/terminal**

When the terminating gateway is H.323, call hunting is triggered based on the following conditions:

- **Destination is unreachable** - If the iServer is unable to establish a TCP connection with the destination gateway over which it sends H.323 setup messages, this triggers the iServer to hunt for an alternate destination. This could indicate that either there is no network route to the destination gateway, or it is temporarily out of service.
- **ISDN cause code in the release complete message** - The presence of any completion code defined as triggering hunting will start the hunt sequence. The iServer ships from the factory with hunting triggered by the codes noted in the list below. This list is user-configurable, as noted in the chapter *Custom configuration* on page 535. Each code is described in Table 19, “Listed ISDN Cause Codes (field #30)”, on page 413.

For details on which cause codes provoke hunting, see “Cause Code Operations” on page 524.

- **H.323 release complete reason** - The presence any of the following release complete cause codes in the release complete message will automatically trigger call hunting.

---

2. *Provided* the **Hunt** option is enabled on the **Billing** tab of the iServerConfiguration window.



- UndefinedReason
- UnreachableDestination
- UnreachableGatekeeper
- TypeInvalidRevision
- GatekeeperResource
- GatewayResource
- CallForwarded
- AdaptiveBusy
- InConf
- RouteCallToMC
- RouteCallToGatekeeper
- FacilityCallDeflection
- ConferenceListChoice
- NewConnectionNeeded
- CalledPartyNotRegistered

If none of the above are present, then dynamic call hunting will be triggered by any of the following CDR errors:

- network error
- no route
- general
- resource unavailable
- destination unreachable
- undefined
- gateway resource unavailable
- disconnect unreachable
- sip internal
- sip service unavailable

### ***Termination entity is SIP***

When the terminating entity is SIP, call hunting is initiated based on the following conditions.

- **Destination is unreachable** - Call hunting is initiated when the iServer concludes that an endpoint cannot be reached. This can happen in two ways:
  - The iServer receives an ICMP unreachable message in response to the INVITE message that it sends out to the destination. This could indicate that there is no network route to the destination, or the destination is temporarily out of service.
  - Outgoing INVITEs are re-transmitted only the number of times controlled by the `siptrans-invitec` global setting. If that limit is reached, the iServer stops trying that endpoint, and initiates hunting to another endpoint.
- **SIP response codes** - Generally, when the iServer receives a Final Response code in reply to one of its own initial INVITE messages, it will initiate call hunting. The iServer ships from the factory with hunting triggered by the response codes indicated in Table 36, “SIP Factory Configuration” on page 532. This list is user-configurable, as described in *Custom configuration* on page 535.

## Intra-carrier call hunting

Upon failing a call to a destination, the iServer initiates hunting procedures. This hunt sequence allows hunting back to the original destination IP address, in case a different route from the failed route may succeed in completing the call. The new route may be bound to the same port or a different port for the same final destination.

An example of the use of this feature is when there are multiple hunts to the same destination with different phone numbers. This is key in scenarios where the destination in question does not support hunting, as in the case of Clarent gateways.

## Call hunting with multiple directory gatekeepers

Every call that hits the iServer goes to the directory gatekeeper with an LRQ, and the gatekeeper sends back an LCF with up to three destinations, consisting of a gateway IP address and a number to dial.

A company may have iServers co-located with directory gatekeepers. Additionally, it may have backup directory gatekeepers at other locations.

When all the gatekeepers refer to the same route server database, they will all return the same three destinations for a given call, so if the three destinations

provided by the first queried directory gatekeeper do not result in a completed call, there is no point in trying the other gatekeepers.

In situations like this, you can prevent call hunting to alternate directory gatekeepers by setting the `stophuntrxlcg` attribute in `servercfg` to *enabled* (1). To enable this attribute, log on to the iServer and enter:

```
nxconfig.pl -e stophuntrxlcg -v 1
```

See *Call hunting with multiple directory gatekeepers* on page 63 for more information.

## Cause Code Operations

### Introduction

This chapter details the MSX's Cause Code (CC) mapping and CC-based hunt logic functionality, and how to administer and customize it.

*Note: The hunting feature as described in this chapter became available with MSX release 2.06c2. Prior to that release, hunting was supported, based on two ISDN Cause Code lists, known as the “short” hunt list, and the “long” hunt list. The long list included all the codes that caused hunting in the short list, plus some additional codes.*

VoIP operations involve communication between multiple call-handling devices, such as IP phones, gateways, the session controller, and so on. Destination (e.g., endpoint) and intermediate (e.g., session control) devices send coded responses back to the originating device, so that it can take appropriate action. For example, a far-end response indicating a busy destination device is returned to the originating device so that it can notify the caller with a “busy signal.”

One mechanism by which this kind of information is communicated is known in H.323 as an *ISDN cause code* (or *CC*), and in SIP as a *response code*.<sup>1</sup>

### Mapping

Because some devices that receive CCs are set up to understand only a subset of all possible codes for the protocol they support, a received code-to-sent code mapping can translate a received code to one that the device receiving it knows. This same principle also applies to codes that originate within the MSX (the *intermediate* codes).

H.323 has one set of intrinsic codes, and SIP (of course) has a completely different set. NexTone MSX's IWF functionality includes CC handling. Codes are mapped both within a protocol<sup>2</sup> (e.g., H.323 code A → H.323 code B), and across protocols (H.323 ↔ SIP).

---

1. In this chapter both will be referred to generically as cause codes, or CCs, unless speaking particularly of one or the other.

On the MSX, the mapping of cause codes is configurable, according to the needs of each NexTone customer, to accommodate their equipment and practices.

### **Mapping-related CDR fields**

Four fields in the CDR structure relate to ISDN cause codes and SIP response codes for failed call setups. These are:

- isdn-cause-code (field 30)
- sip-dest-respcode (field 45)
- original-isdn-cause-code (field 60)
- sip-src-respcode (field 75)

Fields 45 and 60 relate to call leg 2; fields 30 and 75, to leg 1.

Fields 30 and 60 relate to H.323; fields 45 and 75, to SIP.

These four fields are described in the chapter, *Billing and CDR Processing* on page 393. With respect to CC mapping, the MSX processes them as follows:

1. Place the code received from leg 2's endpoint into the CDR field corresponding to leg 2's protocol.
2. Map the code received in step 1 to the corresponding *other* protocol, and place that into leg 2's other protocol field.
3. Map the code received in step 1 to the code to be sent to leg 1 (for leg 1's protocol) and place it in leg 1's *same* protocol field.
4. Map the code resulting from step 3 to the corresponding *other* protocol, and place that into leg 1's other protocol field.

Therefore, these fields have the following relationships:

- For an H.323-H.323 call released at leg 2, with ISDN cause code *X*:
  - Field 60 contains *X*
  - Field 45 contains the SIP response code to which *X* maps
  - Field 30 contains *X* (or mapped code *Y* if a mapping is enabled, as sent to leg 1)
  - Field 75 contains the SIP response code mapped from the CC in field 30

---

2. Note that SIP → SIP CC mapping is available, but limited. If you need this function, contact NexTone Support for evaluation and assistance.

- For a SIP-H.323 call released at leg 2, with ISDN cause code *X*:
  - Field 60 contains *X*
  - Field 45 contains the SIP response code to which *X* maps
  - Field 75 contains the mapped SIP response code *Y* sent on leg 1
  - Field 30 contains the mapped ISDN CC for field 75's response code
- For a SIP-SIP call released by leg 2, with SIP response code *X*:
  - Field 45 contains *X*
  - Field 60 contains the ISDN CC to which *X* maps
  - Field 75 contains the SIP response code sent on leg 1 (which always will be *X* unless SIP-SIP mapping from leg 2 to leg 1 has been configured. See footnote 2 above.)
  - Field 30 contains the ISDN CC to which field 75 maps
- For an H.323-SIP call released by leg 2, with SIP response code *X*:
  - Field 45 contains *X*
  - Field 60 contains the ISDN CC to which *X* maps
  - Field 30 contains the mapped ISDN CC *Y* sent on leg 1
  - Field 75 contains the SIP response code mapped from field 30

### **Hunting**

In addition to CC mapping, call hunting<sup>3</sup> behavior is also based on CCs, and the MSX can be configured to control hunting for a particular CC received from the call's egress leg.

### **Factory default settings**

As shipped from NexTone, an MSX is configured to map cause codes and hunt according to the information presented in the three tables included below. In the first two tables, every receivable code defined for a protocol (H.323 in Table 35, and SIP in Table 36) is listed in the table's left-most column. The corresponding hunt behavior and mappings are shown in the other columns. Each receivable code has two potential mappings:

- Received code-to-sent code in the *same* protocol, and

---

3. See *Dynamic Call Hunting* on page 519 for details of how hunting works.

- Received code-to-sent code in the *other* protocol.

Cause codes that have specific meaning to (and provoke specific behavior from) the MSX are listed in the H.323 and SIP chapters of this book. For descriptions of others, please consult that protocol's specification (noting that Sent H.323 Codes listed in Table 36 as "-1" correspond to Received SIP Codes that are unused or undefined. As such, those codes may not appear in the SIP protocol specification).

The third table, Table 37, lists the MSX's internal, intermediate condition codes, and the ISDN and SIP cause codes to which they map.

### ***Received H.323 code mapping and hunt behavior***

Table 35 presents the hunt behavior for each received H.323 code, along with the sent codes (one H.323/ISDN and one SIP for IWF calls) to which each possible received H.323 code maps. Valid ISDN CC values range from 0 (zero) to 127, inclusive. Valid SIP response code values range from 400 to 699 inclusive.

***Note: For readability, large blocks of repeating data have been condensed into one row representing a range. In such cases, the sent code is the same as the received code, and the Sent H.323 Code reads "(rec'd.)".***

**Table 35. H.323 Factory Configuration**

Received H.323 Code	Hunt?	Sent H.323 Code	Sent SIP Code
0	No	0	500
1	No	34	404
2	Yes	34	404
3	Yes	3	404
4	No	4	500
5	No	5	500
6	Yes	6	500
7-16	No	(rec'd.)	500
17	No	17	486
18	No	34	480
19	No	19	480
20	No	20	480
21	Yes	34	403
22	No	22	410
23	No	23	500
24	No	24	500
25	No	25	500
26	No	26	404
27	Yes	34	404
28	No	34	484
29	Yes	34	501
30	Yes	30	500
31	Yes	31	404
32	No	32	500



Received H.323 Code	Hunt?	Sent H.323 Code	Sent SIP Code
33	No	33	500
34	Yes	34	503
35	No	35	500
36	No	36	500
37	No	37	500
38	Yes	38	503
39	No	39	500
40	No	40	500
41	Yes	41	503
42	Yes	42	503
43	Yes	43	500
44	Yes	44	500
45	No	45	500
46	No	46	500
47	Yes	47	503
48	No	48	500
49	Yes	49	500
50	Yes	34	500
51	No	51	500
52	No	52	500
53	No	53	500
54	No	54	500
55	No	55	403
56	No	56	500
57	Yes	57	403

Received H.323 Code	Hunt?	Sent H.323 Code	Sent SIP Code
58	Yes	58	501
59	No	59	500
60	No	60	500
61	No	61	500
62	No	62	500
63	Yes	34	500
64	No	34	500
65	Yes	65	501
66	Yes	66	500
67	No	67	500
68	No	68	500
69	Yes	69	500
70	Yes	70	500
71-78	No	(rec'd.)	500
79	Yes	79	501
80	No	80	500
81-85	Yes	(rec'd.)	500
86	No	86	500
87	No	87	503
88	Yes	88	400
89	No	89	500
90	No	90	500
91	Yes	91	500
92	No	92	500
93	No	93	500

Received H.323 Code	Hunt?	Sent H.323 Code	Sent SIP Code
94	No	94	500
95	Yes	95	400
96-101	Yes	96	500
102	Yes	102	408
103-110	No	(rec'd.)	500
111	Yes	111	400
112-126	No	112	500
127	Yes	127	500

### ***Received SIP code mapping and hunt behavior***

Table 36 presents the hunt behavior, along with two sent codes (one H.323/ISDN for IWF calls and one SIP) to which each possible received SIP code maps. Valid SIP response code values range from 400 to 699 inclusive, though some SIP codes in this range are unused or undefined; and are therefore shown in the table as “-1”. Valid ISDN CC values range from 0 (zero) to 127, inclusive.

***Note:*** For readability, large blocks of repeating data have been condensed into one row representing a range. In such cases, the sent code is the same as the received code, so the Sent SIP Code reads “(rec'd.)” in the table.

**Table 36. SIP Factory Configuration**

Received SIP Code	Hunt?	Sent H.323 Code	Sent SIP Code
400	Yes	127	400
401	No	57	401
402	Yes	21	402
403	No	57	403
404	Yes	1	404
405	Yes	127	405
406	Yes	127	406
407	No	21	407
408	Yes	102	408
409	Yes	41	409
410	Yes	1	410
411	Yes	127	411
412-479	No	-1	(rec'd.)
480	Yes	18	480
481	No	127	481
482	Yes	127	482
483	Yes	127	483
484	Yes	28	484
485	Yes	1	485
486	No	17	486
487	Yes	127	487
488	Yes	127	488
489-499	No	-1	(rec'd.)
500	Yes	41	500

Received SIP Code	Hunt?	Sent H.323 Code	Sent SIP Code
501	No	79	501
502	Yes	38	502
503	Yes	63	503
504	Yes	102	504
505	Yes	127	505
506-579	No	-1	(rec'd.)
580	Yes	47	580
581-599	No	-1	(rec'd.)
600	Yes	17	600
601	No	-1	601
602	No	-1	602
603	Yes	21	603
604	Yes	1	604
605	No	-1	605
606	Yes	58	606
607-699	No	-1	(rec'd.)

### **Intermediate cause codes**

If the session controller needs to notify an originating device of a condition that affects a call, it does so with codes that originate within the controller itself. An example of this case would be if the customer is already using the number of ports available to it, so that a new call cannot be set up. The internal error is “no-ports”, and this maps to an ISDN cause code, and also to a SIP cause code.

The mappings of the errors to the *servercfg* attributes that set the values sent, and the default values for the sent H.323 codes, as defined in the shipped *servercfg* table, are given in Table 37. Note that the SIP codes are not set in the *servercfg* table.

**Table 37. Intermediate Codes, Factory Configuration**

Intermediate Error	servercfg Attribute	Sent H.323 Code	Sent SIP Code
abandoned	err2isdn-abandoned	16	500
user-blocked	err2isdn-user-blocked	21	403
user-blocked-at-dest	err2isdn-user-blocked-at-dest	21	403
no-route	err2isdn-no-route	34	404
no-route-at-dest	err2isdn-no-route-at-dest	3	404
busy	err2isdn-busy	17	486
resource-unavailable	err2isdn-gw-resource-unavailable	47	480
invalid-phone	err2isdn-invalid-phone	28	484
network-error	err2isdn-network-error	34	503
no-ports	err2isdn-no-ports	34	503
general-error	err2isdn-general-error	16	503
dest-unreach	err2isdn-dest-unreach	27	480
undefined	err2isdn-undefined	31	503
no-bandwidth	err2isdn-no-bandwidth	34	503
h245-error	err2isdn-h245-error	127	503
incomp-addr	err2isdn-incomp-addr	28	484
local-disconnect	err2isdn-local-disconnect	16	403
h323-internal	err2isdn-h323-internal	41	503
h323-proto	err2isdn-h323-proto	41	503
no-call-handle	err2isdn-no-call-handle	41	503
gw-resource-unavailable	err2isdn-gw-resource-unavailable	47	503
fce-error-setup	err2isdn-fce-error-setup	34	500
fce-error	err2isdn-fce-error	34	500
fce-no-vports	err2isdn-fce-no-vports	34	500

Intermediate Error	servercfg Attribute	Sent H.323 Code	Sent SIP Code
no-vports	err2isdn-no-vports	34	503
h323-maxcalls	err2isdn-h323-maxcalls	34	503
msw-invalid-epid	err2isdn-msw-invalid-epid	31	503
msw-routecallgk	err2isdn-msw-routecallgk	31	503
msw-notreg	err2isdn-msw-notreg	31	503
hairpin	err2isdn-hairpin	91	500
shutdown	err2isdn-shutdown	31	500
switchover	err2isdn-switchover	31	500
disconnect-unreach	err2isdn-disconnect-ureach	102	480
dest-relcomp	err2isdn-dest-rel-comp	31	500
temporarily-unavailable	err2isdn-temporarily-unavailable	17	480
dest-gone	err2isdn-dest-gone	22	500
dest-timeout	err2isdn-dest-timeout	42	503
reject-route	err2isdn-reject-route	34	404

To change the factory default mapping, edit the value of the appropriate servercfg attribute. To edit a value, log into the iServer and enter:

```
nxconfig.pl -e error attribute -v value
```

where error attribute and value is the new integer value to assign to the code.

## Custom configuration

As shipped from NexTone, hunting and mapping behavior is as described in the tables above. This section details how to customize the mapping and hunting behaviors defined in the preceding tables.

### Configuration files

The behavior of this feature is controlled by a binary file that NexTone generates from a human-readable, editable text file. To change the mapping

and hunting behaviors, you edit a copy of the text file and submit it to NexTone Support, who then generate a new binary file for you. Finally, you set the `usecodemap` attribute in `servercfg` so that it points to the new binary, and configure endpoints to hunt and map according to that new definition. These steps are detailed below.

Two basic configuration files for this feature are supplied from the factory. These are named:

- `codemap_001.dat` (generated from `codemap_001.txt`), and
- `codemap_002.dat` (generated from `codemap_002.txt`)

**Note:** *Some codemap files with iteration numbers beyond 001 and 002 are also installed during a standard MSX upgrade or installation. These are customized in undocumented ways, and should therefore not generally be used as a basis for derivative configurations.*

### **001 vs. 002**

The two files named above differ as follows:

- The 001 file provides functionality corresponding to the previous *short hunt list* used in prior releases. This “short” list contained a group of H.323 CCs that would initiate hunting.
- The 002 file provides functionality corresponding to the previous *long hunt list* used in prior releases. The difference is that in this longer list, the following additional H.323 CCs were defined as resulting in a hunt: 0 (zero), 1, 4, 7, 18, 20, 26, 28 and 86. This file is otherwise the same as the 001 file.

Only one configuration file can be active at a time; the active one is the one pointed to in the `servercfg` table’s `usecodemap` attribute. You set this as described *Setting the configuration file to use* on page 538.

**Note:** *Take care not to delete or otherwise overwrite the supplied 001 and 002 files.*

### **Creating a custom configuration**

To customize the mapping and hunting behaviors:

1. Create a text file defining the new behavior (copy and modify a supplied 001 or 002 text file, or another you’ve already created, and save it with a new numeric iteration, up to 999), and contact NexTone Support to arrange for the binary conversion. Send the file per their instructions.
2. NexTone generates a `.dat` binary file from it, and sends that back to you.



3. Place the new `codemap` file on the MSX in its `/usr/local/nextone/bin` directory.
4. Edit the `codemap servercfg` attribute, as described in “Setting the configuration file to use”, below.
5. Enable CC mapping on each H.323 endpoint that will use the new behavior (see *Endpoint configuration* on page 538).

#### **Code map file contents**

The code map files consist of two sections, the first with extensive usage comments, the second with two blocks of hunting and mapping data. The first block of data is for received ISDN (H.323) cause codes, and the second for received SIP response codes. Edit the file using your preferred text editor.

The fields in the two data blocks are arranged in rows and columns/fields as follows:

**Field 1:** Integer **received code** for that protocol

**Field 2:** **Hunt behavior**; valid values are case-insensitive *no* or *yes*

**Field 3:** Mapped, integer **sent code** in the **same** protocol

**Field 4:** Mapped, integer **sent code** in the **other** protocol

#### **File content rules**

The H.323 and SIP blocks differ in what fields can be edited in them. Here are the rules:

- In neither block should you change the contents of field #1. This is the “from” field that represents the received code. This list is and should remain contiguous numbers.
- In both blocks you may change the hunt behavior (field 2).
- In the H.323/ISDN block, you may change the mapped-to (sent) code in the same protocol (field 3). Since SIP-to-SIP mapping is not generally supported in this release, you should not change the *sent* code in the SIP block (see footnote 2 on page 525).
- Never change a “-1” value in field 3 of the SIP block.
- In both blocks you may change the mapped-to (sent) code in the *other* protocol (field 4).
- Whenever changing the contents of fields 3 or 4, take care to enter only valid ISDN (field 3) or SIP (field 4) codes.

### Setting the configuration file to use

The MSX software comes from NexTone configured to use the included `codemap_001.dat` mapping file. To tell the MSX to use a different binary configuration file, you set the `usecodemap` attribute of `servercfg` to the three-digit sequence number of the codemap file you want to use. For example, to specify the `codemap_002.dat` file, log into the iServer and enter:

```
nxconfig.pl -e usecodemap -v "002"
```

**Note:** *If the number you specify has no corresponding .dat file in /usr/local/nextone/bin, the MSX reverts to the 001.dat version.*

Note that the "002" in the above example is one possible entry; many versions of file could be saved in the directory, and this would point to the one being used, from 001 to 999.

To just hunt on the CCs defined in the original "long hunt list," specify 002 here.

### Endpoint configuration

#### Hunting

Once the MSX is configured to support hunting as described above, all endpoints (both H.323 and SIP) will exhibit the defined hunting behavior without further endpoint-specific configuration.

#### Cause code mapping

CC mapping must be enabled on each H.323 endpoint that will use it. In this release, only H.323-H.323 CC mapping is supported; SIP-SIP response code mapping is not supported.

There are two ways to enable CC mapping on an endpoint:

- Enable it system-wide, by setting the global `mapisdnc` option in `servercfg`
- Enable it just for that endpoint from within RSM Console

#### **Global CC enable**

To enable CC mapping system-wide, set the `mapisdnc` `servercfg` table attribute to 1 (enabled), by logging into the iServer and typing:

```
nxconfig.pl -e mapisdnc -v 1
```

**Note:** *Even if the endpoint's configuration does not show ISDN CC Mapping as checked, if mapping is enabled globally, mapping is performed.*

***Endpoint-specific CC enable***

To configure a specific H.323 endpoint for CC mapping, go to that endpoint's H.323 Protocol Parameters panel (**Modify H.323** [endpoint type] panel → **Protocol** tab → **H.323 Configure** button), shown (partially) in Figure 94. Select the **ISDN CC Mapping** option box.

**Figure 94. Setting ISDN CC Mapping (Partial View)**

The screenshot shows the 'H.323 Protocol Parameters' dialog box. The 'H.323' tab is selected. The following fields are visible:

- H.323 Id: Cust\_A
- RAS Port: 1719
- Q.931 Port: 1720
- Calling Party Number Type: Pass(Default)
- Called Party Number Type: Pass(Default)
- Layer1 Protocol: System Default
- Info Transfer Cap: default
- ☐ Q.931 Display
- ☐ Force H.245
- ☐ H.245 Address in Connect
- ☒ ISDN CC Mapping
- ☐ PI On FastStart
- Remove from TCS
- RFC 2833: default

At the bottom are 'OK' and 'Cancel' buttons.

Click **OK** twice, and the endpoint performs the mapping indicated in the currently-defined `codemap` file.

# **Part 5:**

## **Reference Appendices**

# A

## GLOSSARY

This appendix provides a basic glossary for some of the terms, acronyms, and abbreviations used in this book.

Term	Definition / Description
<b>ACL</b>	Access Control List. In the VoIP context, an ACL is a table listing originating endpoints from which calls will be accepted by a particular receiving endpoint, or which a firewall will allow to pass into the private network it protects.
<b>A-Law</b>	The 16-to-8 bit loss-less audio compression algorithm used outside of North America. See also “ $\mu$ -Law”.
<b>ANI</b>	Automatic Number Identification. A service that provides the receiver of a telephone call with the number of the calling phone.
<b>ASCII</b>	American Standard Code for Information Interchange. A seven-bit binary encoding of printable and non-printable (control) characters. Most modern computer systems use this code to communicate with each other and the outside world. It is also the character set used from the command line.
<b>ASR</b>	Answer Seizure Ratio. Ratio of the number of successful calls over the total number of outgoing calls from a carrier’s network.
<b>AudioCodes Transcoder module</b>	A rack-mountable cPCI hardware module with a single Ethernet interface. The transcoder allows third party devices to send control commands to it through SIP, MGCP or TPNCP (AudioCodes proprietary). The current release uses TPNCP as the control protocol between MSX and the transcoder.
<b>Bonded interface</b>	The Control LAN can be configured with redundancy, through a Linux technique known as “bonding.” Bonding joins two interfaces logically under a single Linux interface name. The iServer uses a logical interface called <code>bond0</code> for this.

Term	Definition / Description
<b>CAC</b>	Call Admission Control. A graceful approach to handling the situation where adding a new VoIP call would exceed the IP network's available capacity, or a customer's purchased bandwidth.
<b>Calling plan</b>	A set of routes logically grouped under one name.
<b>Carrier</b>	An external call transporting entity that connects to the VoIP network. Carriers are most often "peering partners" of the owner of the VoIP network controlled by the session controller with which NARS is associated.
<b>CDR</b>	Call Detail Record. Contains details of a processed call.
<b>Channel</b>	Refers to resources allocated on a transcoder for one leg of a call. A channel is a bi-directional entity. Attributes of a channel include codec, codec parameters, the RTP/RTCP address of the peer, and TDM bus interface related settings.
<b>CIDR</b>	Classless Inter-Domain Routing. Early IP subnetting techniques were grouped in 8-bit increments, known as Class A (8 bits), Class B (16 bits), etc. CIDR allows 1-bit incremental subnetting just by specifying the number of "1" bits in the subnet mask. (Under this notation, a network address in a block of 4 Class C networks, for example, would be written, "192.168.20.3/22," for example.)
<b>Cisco IOS</b>	IOS is Cisco's network management software suite.
<b>Codec (coder-decoder)</b>	A codec translates analog or digital audio or video signals into alternate formats. Audio codecs use a variety of data compression algorithms and settings to compensate for marginal network service quality, or to maximize voice quality while minimizing bandwidth and processor use. Many different codecs are in use in telephony. MSX supports digital-to-digital audio stream transcoding. Video is not supported.
<b>Codec profile</b>	Captures the codec-related policy for an endpoint through the definition of sets of preferred and prohibited codecs (in decreasing order). The preferred set defines the codecs allowed for RX/TX on the endpoint. The prohibited set defines the codecs barred for RX/TX on the endpoints. These sets are mutually exclusive and can have the values NULL or ALL. Codecs that can be used in a codec profile include vocoders g.711u, g.711a, g729, and g7231. Codec profiles do not describe the capabilities of an endpoint, but instead impose network policies on them.

Term	Definition / Description
<b>Command line</b>	The point of interface into low-level communication with a computer system. Opening a terminal session on a iServer computer results in the system providing a prompt on one line into which a user types commands.
<b>Control LAN</b>	The local area network over which heartbeat packets are sent between an active processor and its peer(s).
<b>CSC</b>	Core Session Controller. Another term for an MSX which is placed at the “core” of a network, and which provides call routing services through it.
<b>CSV</b>	Comma-separated value. A file format used for transferring ASCII data between systems whose native file formats are incompatible. Data in the file takes the form <code>field1,field2,...</code> . Note that the field separator character does not <i>have</i> to be a comma. It can be any ASCII character that doesn't occur in the data itself, such as a semicolon, tilde, etc.
<b>.CTT</b>	A CDR file for non-traditional CDRs (ingress leg start-call, and egress leg start- and end-call).
<b>Database</b>	An ordered collection of information designed for rapid access. The iServer database contains its endpoint registration and configuration information.
<b>Default route</b>	A route used when the destination host or network is not in the routing table.
<b>Dialog box</b>	A browser or application window used to enter and/or modify data, consisting of a combination of text boxes, lists, pull-downs, etc.
<b>DNIS</b>	Dialed Number Identification Service. A telephone service that identifies for the receiver of a call the number that the caller actually dialed. It's a common feature of 800 and 900 lines. If you have multiple 800 or 900 numbers to the same destination, DNIS tells which number was called. DNIS works by passing the DTMF digits to the destination.
<b>DNS</b>	A service that converts a FQDN to an IP address.
<b>Dual Tone Multiple Frequency (DTMF)</b>	A composite, two-tone audio signal used in touch-tone dialing and for telephony in-band signaling, interactive voice response (IVR), and other applications. LBR codecs do not reliably carry DTMF. Circuits using LBR codecs should use the RFC 2833 method for transporting DTMF information.

Term	Definition / Description
<b>Egress-leg</b>	The call leaving the iServer, going to a termination endpoint.
<b>Endpoint</b>	<p>A generic term describing an entity that sends calls to or receives calls from the session controller (iServer). An endpoint can be of various types - SIP gateway, H.323 gateway, SIP proxy server, Softswitch, IP phone etc.</p> <p>Endpoints are the sources or sinks of data. An example could be an interface on a gateway that terminates a trunk connected to a PSTN switch.</p>
<b>ENUM</b>	The Internet Telephone Numbering System—the technology that bridges the public switched telephone network and the Internet. ENUM works as a kind of DNS for IP telephony; it converts an E.164 telephone number to an Internet domain.
<b>ESC</b>	Edge Session Controller. Another term for an MSC, which is placed at the “edge” of a network, and which controls access to it.
<b>Failover</b>	A standby CPU taking processing duties from an active CPU.
<b>FCE</b>	Firewall Control Entity. Software used to configure firewall functions. Combines with NSF to implement media routing.
<b>FQDN</b>	Fully-Qualified Domain Name. The complete alphabetic address, as stored in your DNS, of a server, such as www.nextone.com.
<b>Gatekeeper</b>	An H.323 entity responsible for managing and authorizing calls to one or more H.323 gateways under its control. Gatekeepers are optional nodes that manage endpoints in an H.323 network. The endpoints communicate with the gatekeeper using the RAS protocol.
<b>Gateway</b>	An endpoint on the LAN that provides real-time communications between H.323 terminals on the LAN and other ITU terminals on a WAN or to other H.323 gateways. Gateways allow H.323 terminals to communicate with devices that are running other protocols. They provide protocol conversion between the devices that are running different types of protocols.
<b>gis/gis process</b>	The iServer process responsible for call processing and control.
<b>Glare</b>	A race condition typically induced during call processing where both ends of the call send similar capabilities at the same time.



Term	Definition / Description
<b>H.225</b>	An ITU standard that governs H.225.0 session establishment and packetization. H.225.0 actually describes several different protocols: RAS, use of Q.931, and use of RTP.
<b>H.235</b>	H.235 provides security for the RAS signaling between H.323 endpoints and gatekeepers so that only duly authenticated and authorized endpoints are able to use Gatekeeper resources.
<b>H.245</b>	The call signaling protocol and media stream packetization standard for packet-based multimedia communication systems. Used with the H.323 family of protocols. <i>H.245 Tunneling</i> reduces call-setup time and ensures a more efficient use of network resources. H.245 Tunneling can be used only when both endpoints have this capability. When this is not the case, H.245 negotiation is performed via separate TCP connections.
<b>H.323</b>	The globally accepted standard for packet-switched audio/video/data communication over an IP network. It specifically describes how multimedia communications occur between user terminals, network equipment, and assorted services on local and wide-area IP networks. The H.323 standard uses other standard protocols such as H.245 and H.225.
<b>H.323 proxy</b>	Special types of gateways that relay H.323 calls to another H.323 endpoint. They can be used to isolate sections of an H.323 network for security purposes, to manage quality of service (QoS), or to perform special application-specific routing tasks. An H.323 gatekeeper is required if H.323 proxies are used.
<b>HA</b>	High Availability. A method of minimizing downtime and its effects. Usually involves eliminating single points of failure by providing replication of data and duplication of hardware.
<b>Hairpin</b>	Calls that have the same source and destination gateways.
<b>High-bit-rate (HBR) codec</b>	A codec that provides excellent voice quality and handles complex, non-voice audio well, but requires greater bandwidth than LBR codecs. G.711 $\mu$ -Law and g.711a-Law are two common HBR codecs.
<b>Hunting</b>	Call hunting is forwarding a call to one or more numbers, in sequence, if the original dialed number (a) doesn't pick up within a specified time limit, or (b) is busy or otherwise unavailable.
<b>IETF</b>	The Internet Engineering Task Force

Term	Definition / Description
<b>inform</b>	And SNMP notification with acknowledgment.
<b>infrastructure ENUM</b>	Infrastructure ENUM maps a telephone number into a URI such that the URI maps to a specific point of interconnection to the service provider's network that could enable the originating party to establish communication with the associated terminating party. This URI is separate from any URIs that the end-user who registers his E.164 number in ENUM may wish to associate with that E.164 number.
<b>Ingress-leg</b>	From the perspective of the iServer, this is an incoming call.
<b>IOG</b>	This <i>Installation and Operations Guide</i>
<b>iServer</b>	See MSX. iServer refers to the software that manages the VoIP capability.
<b>iView</b>	See RSM Console.
<b>IWF</b>	The Interworking Function. The iServer's bridge between the H.323 and SIP signaling protocols.
<b>Low-bit-rate (LBR) codec</b>	A codec that compresses data to minimize bandwidth use. The compression algorithms used by several LBR codecs assume voice traffic waveforms. Therefore, LBR codecs cannot reliably carry the composite and modulated waveforms associated with modem and DTMF audio. G.729 and G.723.1 are two well-known voice LBR codecs.
<b>LRQ</b>	Location Request. Information sent by a H.323 gatekeeper such as the iServer to another peer H.323 gatekeeper requesting the location of a certain endpoint.
<b>Management network</b>	A non-service LAN used to remotely monitor, link, and/or control assets in an operations center; this network should be secure and accessible even if the service network is down.
<b>MAC Address</b>	A unique identifier assigned every Ethernet interface, globally. This identifier is used for normal network communication, and on iServers, one of its MAC addresses can also be used to control feature licensing.
<b>MCU</b>	Multipoint Control Unit. An endpoint on the network that allows three or more endpoints to participate in a multipoint conference.

Term	Definition / Description
<b>Media</b>	Content, whether voice, fax, video, or other forms of communication, transmitted between endpoints; as opposed to the other component of VoIP, signaling.
<b>Media device</b>	A logical entity that intercepts media and modifies RTP layer and/or IP layer attributes for each media packet. Depending on the modifications, the device may have to terminate and re-originate the RTP stream and/or simply forward IP packets with modifications. NSF, NSF-NP, NSF-NP2, and transcoder are examples of media devices.
<b>Media routing</b>	The ability to route media through a central network element so that the endpoints involved in the media flow send their media only to this central network element. Media routing enables forced routing through a network “common point,” and hides the endpoints’ addresses (and therefore their identities) from one another.
<b>Media services layer</b>	A layer of abstraction on top of different types of media devices—for example, QoS Router, Transcoder, firewall, and so on—that provides a homogenous service interface to the signaling plane. The media services layer deduces the media devices required to fulfill the service request from signaling and then synchronizes service requests to individual devices.
<b>MGCP</b>	Media Gateway Control Protocol. The Media Gateway Control Protocol is a control and signal standard competing with the older H.323 standard for the conversion of audio signals carried on telephone circuits (PSTN) to data packets carried over the Internet or other packet networks. MGCP is meant to simplify standards for VoIP by eliminating the need for complex, processor-intense IP telephony devices, thus simplifying and lowering the cost of these terminals.
<b>MSCP</b>	Media Services Control Protocol. MSCP is a control protocol used by NexTone and supported on Snowshore/Brooktrout media firewalls.
<b>MIB</b>	Management Information Base. A textual description of managed SNMP objects.
<b>μ-Law</b>	The 16-to-8 bit audio compression algorithm used in North America. See also “A-Law”.

Term	Definition / Description
<b>MPLS</b>	Multi-Protocol Label Switching. A short fixed-length label represents an IP packet's full header. Subsequent routing decisions are based on the MPLS label and not the original IP address. Supports more complex services and QoS.
<b>MSX</b>	The NexTone Multiprotocol Session Exchange (MSX). A VoIP session controller, comprising a SIP/H.323 interworking element and Firewall Control Entity (FCE) on a single platform deployed at the network edge.
<b>NAS</b>	Network Access Server. A server used to authenticate users to a RADIUS server. The iServer acts as an NAS when configured to use a RADIUS server for authentication, authorization and accounting.
<b>NAT</b>	Network Address Translation. A system, often implemented in a router or firewall, used commonly for security and/or to allow many devices on a private network to share one internet public address.
<b>NIC</b>	Network Interface Card
<b>NMS</b>	Network Management System. A combination of hardware and software used to monitor and administer a network.
<b>Node</b>	<ol style="list-style-type: none"> <li>1. Generically, any network endpoint, which may include an iServer, LAN router, gateway, etc.</li> <li>2. A point of iServer service. In the case of a single machine system, this is the iServer box; in a redundant system, both machines together are known as one "service node."</li> </ol>
<b>notification</b>	An SNMP <i>trap</i> , or <i>inform</i>
<b>NSF</b>	NexTone Session Filter. A packet filter that provides isolation between the two legs of a call, and hides the topology of the networks connecting to that common point. The iServer integrated with the NSF performs the functions of the "media router."
<b>OS</b>	Operating System. For the iServer, this is Sun Microsystems' "Solaris for x86" (for releases 3.x and prior), or NexTone Linux (a.k.a. <i>NxLinux</i> ; releases 4.x and subsequent).
<b>OSP</b>	Open Settlement Protocol. A mechanism by which CDRs can be reported to a central place with settlement between service providers.

Term	Definition / Description
<b>PDD</b>	Post dial delay. The amount of time that a caller or calling endpoint has to wait before getting any indication that the call is going to be completed or not. This information is also captured in the iServer CDRs.
<b>Peering Partner</b>	A third-party company that works with the owner of the VoIP network to complete the transport into and/or out of their network.
<b>PM</b>	Process Manager. A process that watches and controls other NexTone processes.
<b>Pool</b>	In realm-based routing, a pool is a named set of realm media firewall resources (addresses and ports). See <i>Setting up media devices and media routing pools</i> , on page 297.
<b>Pull-down</b>	A list of values in a GUI accessed by clicking on a downward-pointing arrow.
<b>Q.931</b>	A layer 3 call signaling and setup protocol, originally for ISDN, now generalized for VoIP call setup and breakdown. It is roughly comparable to TCP in the Internet protocol stack.
<b>RADIUS</b>	Remote Authentication Dial-In User Service. A widely-used protocol for centralized authentication and accounting. iServer includes a RADIUS client to facilitate delivery of accounting information to third-party servers in XA3+ format, and for authentication of user credentials for certain SIP message types.
<b>RAS</b>	Registration, Admission and Status. Part of the H.225 protocol, it is the call setup protocol used in the H.323 standard.
<b>Realm</b>	In an iServer-controlled network, a realm gives a unique identity to a private network.
<b>Realm Hopping</b>	An application where, for a single call, a single iServer looks like multiple, logically-cascaded proxies, with each serving a different realm.
<b>Redirect Server</b>	An entity that can send a SIP 3xx, H.225 ACF or LCF to the iServer in response to an INVITE, H.225 ARQ or LRQ, respectively.
<b>reg ID</b>	Registration Identification. A credential for an endpoint in the iServer database. This entity is generated by the Network Administrator, and must be unique to an iServer database. Each reg ID can be split into sub-entities called uports.

Term	Definition / Description
<b>Reject route</b>	A type of sub-route defining an exception to a higher-level route. An example would be a route covering all exchanges within an area, with an associated reject route for one exchange within that area requiring different routing from the rest of that area's exchanges.
<b>RFC 2833</b>	A standard developed by the Internet Engineering Task Force (IETF) that provides reliable transportation of DTMF over LBR codec circuits. RFC 2833 specifies a means for converting the audio-based RTP DTMF packets into data-coded RTP packets, and vice versa.
<b>RRQ</b>	Registration Request. Information sent by a dynamic endpoint to the H.323 iServer. Comes in two flavors: normal, and "lightweight." The endpoint periodically sends lightweight RRQs to the iServer to say that it is still active. "Normal" (i.e., full) RRQs actually register an endpoint with the iServer.
<b>RMA</b>	Realm Media Address. The IP address through which a realm sends and receives its media traffic. Each realm has its own unique RMA.
<b>RSA</b>	Realm Signaling Address. The IP address to which signaling messages are sent, so the iServer knows realm in which the traffic originated. Each realm has its own unique RSA.
<b>RSM Console (formerly iView)</b>	iView is NexTone's provisioning and configuration application that uses a graphical user interface (GUI) to provide an easy and user friendly medium for most network provisioning tasks.
<b>RTP</b>	Real-Time Protocol. Standard defined by the IETF for the transmission of media (such as voice, video and fax) over a packet network.
<b>SDL</b>	Service Description Language (SDL) provides an XML-based grammar for describing the capabilities of Web Services. Used with SOAP.
<b>Service network</b>	The network handling call signaling and/or media traffic for customers and users; contrast with <i>Management network</i>
<b>Session</b>	A session is defined by the IETF as "an exchange of data between an association of participants." Practically, in the context of iServer, a session is an activity to set up, conduct or tear down a telephone call, or to facilitate doing so, such as with device registrations or contact list exchanges.

Term	Definition / Description
<b>Session Controller</b>	A system that keeps track of VoIP connections and traffic flow, and produces call detail records describing the activity. The MSC is a session border controller.
<b>Session limit</b>	The ability of the iServer to limit the number of calls either originating from or terminating to an endpoint.
<b>Signaling</b>	That part of the network traffic that sets up sessions or connections over which media (i.e., the call's contents) travel.
<b>SIP</b>	Session Initiation Protocol. A signaling protocol for Internet conferencing, telephony, presence, events notification and instant messaging.
<b>SIP Privacy</b>	A feature that affects caller identification information content and display. For complete information, see <i>SIP privacy</i> , on page 237.
<b>SIP Proxy Server</b>	An intermediate system that processes SIP messages from SIP endpoints. Proxy servers can be stateless or stateful.
<b>SNMP</b>	Simple Network Management Protocol. A standard communication protocol and information base for querying and controlling devices on a network, and receiving notifications.
<b>SOAP</b>	Simple Object Access Protocol. A lightweight, XML-based protocol for the exchange of information or messages in a decentralized, distributed environment. Used with SDL.
<b>Stateful</b>	A <i>stateful</i> device is one that tracks the condition, or status, of something, such as a proceeding telephone call, with start and end. A stateful iServer, for example, is able to generate CDRs because it “knows” when a call began and ended, and therefore can provide a call duration value. A stateless iServer cannot do this, because it does not know the status of the connection once it is made.
<b>Stateless</b>	A stateless device is one that does not track the status of something, such as a proceeding telephone call. See <i>Stateful</i> .
<b>Streams</b>	Streams provide a mechanism to channel traffic from one source to a specified set of destinations.
<b>Switchover</b>	Movement of call processing duties from a “primary” processor to its designated “standby.”

Term	Definition / Description
<b>TCS</b>	Terminal Capability Set. During H.245 call set-up, the TCS contains a list of capabilities that the terminal supports, to facilitate the negotiation process between terminals.
<b>Template Route</b>	A route used to create other routes. Adaptive Fax Routing (AFR) utilizes template routes. Currently, iServer only supports dynamic route creation based on template routes, as with AFR. Setting the "template" bit directs the iServer not to route calls with this route, because it's only intended as a starting point for "true" routes.
<b>Terminal</b>	H.323 Terminals are the client endpoints on the LAN that provide realtime, two-way communications. They can be realized either as SW-Clients running on a PC or workstation, or as dedicated HW-devices such as "IP phones." All terminals must support voice communications; video and data are optional.
<b>Threshold</b>	A limit for a health parameter that, if exceeded, generates an event.
<b>Trap</b>	An SNMP notification.
<b>Transcoder</b>	A hardware device with digital signal processors (DSPs) to transcode between different codecs. MSX integrates with the AudioCodes transcoder module. For more information on AudioCodes transcoders, refer to specifications for AudioCodes' <i>Mediant 3000</i> (board number TP_6310) and <i>Mediant 2000</i> (board number TP_1610) on <a href="http://www.audiocodes.com">http://www.audiocodes.com</a> .
<b>Transcoding</b>	The process of changing codecs or codec attributes for a media stream.
<b>TPNCP</b>	TrunkPack Network Control Protocol. TPNCP is an AudioCodes proprietary protocol used by the MSX to control and communicate with AudioCodes' transcoding devices.
<b>Trunk</b>	A logical grouping of multiple call-carrying circuits, commonly used as a transmission channel between two network nodes or switching centers. Standard trunks carry 24 (SONET) or 30 (SDH) DS0 calls, but they can be sub-divided.
<b>U-Call Flow</b>	A type of transcoding configuration on MSX with hardware (NSF-NP) media routing that hides the transcoder behind the MSX. In this setup, ingress and egress ports exchanged in signaling SDP are on the firewall. On the backend, redirects in the firewall forward media to the transcoder.



Term	Definition / Description
<b>Uport</b>	A uport is a mechanism in the session controller to allow common properties to be created and managed on an endpoint specified by a reg ID. Uports are virtual, in that they do not exist on the physical device they represent.
<b>URI</b>	Uniform Resource Identifier. A short string that identifies resources in the web: documents, images, downloadable files, services, electronic mailboxes, and other resources. They make resources available under a variety of naming schemes and access methods such as HTTP, FTP, and Internet mail addressable in the same simple way. Also known as a "URL" ("L" for Locator).
<b>URL</b>	See <i>URI</i> .
<b>User ENUM</b>	User ENUM in e164.arpa allows end-users to link either existing E.164 phone numbers or phone numbers assigned specifically for this purpose to applications reachable via URIs on the Internet. The decision to request the domain associated to the E.164 number (opt-in) and to fill the domain with resource records of choice is with the end-user. If an existing E.164 number is used, the end-user must prove the right to use this number with the request of the associated domain.
<b>VIP</b>	Virtual IP address. A logical IP address managed jointly by two or more iServer systems in a redundant cluster. The VIP is mapped to a physical network interface on the active iServer/C server. When a switchover occurs and backup system transitions to primary, the backup system assumes control of the VIP and begins providing service.
<b>VLAN</b>	Virtual LAN. A group of devices on one or more LANs that are configured (using management software) so that they can communicate as if they were attached to the same wire, when in fact they are located on a number of different local or remote LAN segments.
<b>vport</b>	A "virtual port" that exists when a complete connection is set up between a point of network ingress and its corresponding point of egress. Think of it as a logical pipe through which call traffic flows, which is set up for a call, and torn down when the call is over.
<b>Watchdog</b>	A timer that is reset at regular intervals by the process or device that it is "watching", to prevent it from expiring. Timer expiration generates a fault.

Term	Definition / Description
<b>White space</b>	The ASCII code set (see <i>ASCII</i> ), provides a number of characters that are visible only by the effect they have on the characters to which they are adjacent, such as spaces and tab characters. Certain other invisible characters may also be considered white space, such as those produced by holding down the <Ctrl> key and striking another character. A safe rule is that any keystroke not producing a normal, printing character should be considered white space.

## THE COMMAND LINE INTERFACE

---

You can perform all iServer configuration and administration tasks either from RSM Console (or RSM Lite), or from the iServer's Command Line Interface (CLI). Refer to RSM Console online help for the detailed procedure to perform iServer tasks from RSM Console. This chapter lists each task that can be performed from the CLI with its corresponding command. To use the CLI, you must `ssh` into the iServer as `root`.

Note: While the CLI resides in the `/usr/local/nextone/bin` directory, CLI commands can be run from any directory, if the iServer's `$PATH` environment variable includes that directory (by default, they all do). If a particular machine is not set up that way, just `cd` to `/usr/local/nextone/bin`, and enter the command as `./cli` instead of just `cli`.

### GENERAL COMMAND USE RULES

---

This section provides a few general rules for using CLI commands.

#### ***Overall cli command syntax***

All CLI commands begin with the name of the command interpreter, `cli`. For example:

```
cli iedge edit regid uport parameter setting
```

- ◆ The keyword following the interpreter's name (`cli`) is the command class. In this example, it is `iedge`, the class of commands used to set endpoint parameters.
- ◆ The next keyword tells the class what kind of operation to perform, such as `add` new information or `edit` or `delete` existing information.
- ◆ The remainder of the command provides parameters specific to the operation, such as the name of an endpoint to add or delete, a specific `uport` to apply the operation to, and so on.

#### ***IPv4 addresses in place of registration IDs***

Some CLI commands take IPv4 addresses as keys instead of Registration IDs. These are typically the `cli iedge` commands. Note that `cli iedge add` still requires a regid (Registration ID) as an argument. The following example shows the syntax of these commands:

```
cli iedge edit ip4:204.190.208.23 0 ...
```

Note that here, instead of specifying the regid of the endpoint, the IPv4 address is specified using the `ip4:` prefix.

## Addition of prefix to destination endpoints

To simplify routing structures and potentially avoid the necessity of duplicating routes for multiple prefixes, the iServer supports addition of prefixes at the destination endpoint. This allows a set of common and normalized routes to take on additional prefixes (for the DNIS) based on the endpoint's regid and uport combination.

The CLI command for specifying this is as follows:

```
cli iedge edit Cyprus-04 1 ogp 24#
```

In the example above, the regid of the endpoint is "Cyprus-04" and the uport is "1". The prefix is specified as "24#". Once this is provisioned, all calls placed from the iServer going to the regid:uport combination will be prefixed with the specified string. The maximum length of the string is 23 digits/characters.

## ISERVER COMMANDS

Table B-1 lists the CLI commands available to the iServer user.

**Table B-1. iServer Commands**

Task	Command	Variables / Notes
<p><i>Note: For commands in this table:</i></p> <ul style="list-style-type: none"> <li>• <i>Items shown in plain, monospace type are part of the command syntax, and must be typed as shown. For example, <code>cli iedge add</code> is entered exactly that way.</i></li> <li>• <i>Items shown underlined, such as <u>regid</u>, are values supplied by the user such as an endpoint's registration ID.</i></li> <li>• <i>Multiple items in square brackets, i.e., [ ], are optional.</i></li> <li>• <i>Multiple items in curly brackets, i.e., { }, are required; you must enter one of them for the command to work.</i></li> <li>• <i>A vertical bar,  , denotes multiple choices, such as [yes   no] (meaning "yes or no").</i></li> </ul>		
Create a new codec profile	<code>cli cdc add <u>codec_profile_name</u></code>	<u>codec_profile_name</u> = name of the codec profile you want to add. Maximum length is 31 characters.

Task	Command	Variables / Notes
Delete an existing codec profile	cli cdc delete <u>codec_profile_name</u>	<u>codec_profile_name</u> = name of the codec profile you want to delete.
Edit existing codec profile characteristics	cli cdc edit <u>codec-profile-name</u> [prefer <u>codec1</u> , <u>codec2</u> ,... ] [prohibit <u>codec1</u> , <u>codec2</u> ,...]	<u>codec_profile_name</u> = name of the codec profile you want to edit. Enter the codecs in priority order, as a list separated by commas but no spaces.  A list of valid codecs can be obtained by entering cli cdc edit.
Show all codec profiles defined	cli cdc list	
Show codec profiles currently in the iServer's active cache	cli cdc cache	
Add an endpoint to the network	cli iedge add <u>regid</u> <u>low</u> [- <u>high</u> ]	<u>regid</u> = registration ID of the endpoint <u>low</u> = first port number in a range of ports <u>high</u> = last port number in a range of ports
Add VPN(s)	cli iedge vpns <u>regid</u> <u>low</u> [- <u>high</u> ] <u>vpn-name</u> [ <u>vpn-name</u> ] ...	<u>regid</u> = registration ID of the endpoint <u>low</u> = first port number in a range of ports <u>high</u> = last port number in a range of ports <u>vpn-name</u> = name of one or more VPNs under which the endpoint is configured

Task	Command	Variables / Notes
Add phone numbers	cli iedge phones <u>regid</u> <u>low</u> [- <u>high</u> ] <u>phone</u> [ <u>phone</u> ] ...	<u>regid</u> = registration ID of the endpoint <u>low</u> = first port number in a range of ports <u>high</u> = last port number in a range of ports <u>phone</u> = phone number for the endpoint
Add email addresses	cli iedge email <u>regid</u> <u>low</u> [- <u>high</u> ] <u>email</u> [ <u>email</u> ] ...	<u>regid</u> = registration ID of the endpoint <u>low</u> = first port number in a range of ports <u>high</u> = last port number in a range of ports <u>email</u> = email address for the endpoint
Add zones	cli iedge zone <u>regid</u> <u>low</u> [- <u>high</u> ] <u>zone</u> [ <u>zone</u> ] ...	<u>regid</u> = registration ID of the endpoint <u>low</u> = first port number in a range of ports <u>high</u> = last port number in a range of ports <u>zone</u> = name of the zone under which the endpoint is configured
Delete an endpoint from the network	cli iedge delete <u>regid</u> <u>low</u> [- <u>high</u> ]	<u>regid</u> = registration ID of the endpoint <u>low</u> = first port number in a range of ports <u>high</u> = last port number in a range of ports
Find a specified endpoint in the database, using its registration ID and port number	cli iedge find <u>regid</u> <u>uport</u>	<u>regid</u> =registration ID of the endpoint <u>uport</u> =port number to find

Task	Command	Variables / Notes
Locate and list an entry in the database based on its phone number, regid/uport, IP address, URL or realm	<code>cli iedge lkup { <u>phone</u>   <u>regid</u> [<u>uport</u>]   <u>ip-address</u>   <u>url</u>   <u>ip-address</u> [<u>realm</u>] }</code>	<u>phone</u> = a telephone number <u>regid</u> = endpoint registration ID <u>uport</u> = optional uport number for the regid <u>ip-address</u> = endpoint IP address <u>url</u> = endpoint's SIP URL <u>realm</u> = optional realm name to search in.
List all entries in the database	<code>cli iedge list &gt; <u>output file name</u></code>	The output from this command <i>must</i> be sent to a file.
View the hunt sequences for a certain dialed number	<code>cli iedge hunt { <u>regid</u>   <u>cid:caller-id</u>   <u>realm:realm-name</u> <u>source-ip</u> } [<u>uport</u>]   [<u>ani:ani-no</u>]   [<u>dnis:dnis-no</u>]   [<u>called-no</u>]   [<u>dest-tg</u>]</code>	Enter <code>cli iedge hunt</code> for a list of valid query structures. Output lists potential dest. GWs, in order.
Edit an endpoint's configuration values	<code>cli iedge edit <u>regid</u> <u>low</u> [-<u>high</u>] <u>attrib-name</u> <u>attrib-value</u> [<u>attrib-name</u> <u>attrib-value</u>] ...</code>	<u>regid</u> = endpoint's registration ID <u>low</u> = first port number in a range <u>high</u> = last port number in a range <u>attrib_name</u> is the name of an attribute to set; <u>attrib-value</u> is its setting. Valid attributes and values list out if you enter just <code>cli iedge edit</code> at the command line.
Trace the route of a call from a specified registration ID and port, or from a specified caller ID, or IP address to the specified called number	<code>cli iedge route { <u>regid</u>   <u>cid:caller-id</u>   <u>realm:realm-name</u> <u>source-ip</u> } [<u>uport</u>]   [<u>ani:ani-no</u>]   [<u>dnis:dnis-no</u>]   [<u>called-no</u>]   [<u>dest-tg</u>]</code>	Enter <code>cli iedge route</code> for a list of available query structures.
Assign an endpoint to a CAC igrp	<code>cli iedge edit <u>regid</u> <u>low</u>[-<u>high</u>] igrp [<u>igrp-name</u>   none]</code>	none un-assigns the endpoint from all iEdge groups

Task	Command	Variables / Notes
Set information transfer capability for an endpoint	cli iedge edit <u>regid</u> <u>uport</u> infotranscap [speech   unrestricted   restricted   audio   unrestrictedtones   video   pass   default]	For a description of each option, see Table 44, on page 274.
Control sending of T38 or RFC2833 in H.245 TCS block	cli iedge edit <u>regid</u> <u>uport</u> [deltcst38   deltcs2833] [default   enable   disable]	Option descriptions are in <i>H.245 Capabilities Mode Restriction</i> on page 284
Assign a codec profile to an endpoint	cli iedge edit <u>regid</u> <u>uport</u> cdcprfl <u>codec-profile-name</u>	<u>codec_profile_name</u> = name of the codec profile you want to assign. <u>regid</u> = endpoint's registration ID <u>uport</u> = optional uport number for the regid  To remove a profile from an endpoint, use "" (two contiguous double-quote marks) for <u>codec_profile_name</u>
Disable sending g729 variants	cli iedge edit <regid> <uport> nog729variants [enable disable]	disable instructs the iServer to send g729 only, and not variants of g729, when sending a SIP INVITE from this endpoint. See "Disabling g729 variants for IWF calls" on page 292.
Turn on transcoding for an endpoint	cli iedge edit <u>regid</u> <u>uport</u> transcode [0 1]	<u>regid</u> = endpoint's registration ID <u>uport</u> = optional uport number for the regid  0 = Off 1 = On



Task	Command	Variables / Notes
Set the use of RFC2833 telephone event	<pre>cli iedge edit <u>regid</u> <u>uport</u> 2833capable {yes no}</pre> <pre>cli iedge edit <u>regid</u> <u>uport</u> 2833capable unknown {yes no}</pre>	<p><u>regid</u> = endpoint's registration ID  <u>uport</u> = optional uport number for the regid</p> <p>Yes sets the RFC2833 telephone event to required.</p> <p>Yes sets the RFC2833 telephone event to unknown.</p>
Set a required payload type for an endpoint when it receives RFC2833 (DTMF) information	<pre>cli iedge edit <u>regid</u> <u>uport</u> pt2833 <u>integer</u></pre>	<p><u>regid</u> = endpoint's registration ID  <u>uport</u> = optional uport number for the regid  <u>integer</u> = an integer value between 96 and 127.</p>
Configure an endpoint for T.38 capability	<pre>cli iedge edit &lt;reg-id&gt; &lt;uport&gt; not38support disable</pre>	Sets the endpoint for T.38. Disable is the default setting.
Configure an endpoint for fax pass-through capability	<pre>cli iedge edit &lt;reg-id&gt; &lt;uport&gt; not38support enable</pre>	Sets the endpoint for fax pass-through.
Add a VPN	<pre>cli vpn add <u>vpn-name</u> <u>vpn-id</u> <u>extn-len</u> [<u>vpn-group</u>]</pre>	<p><u>vpn-name</u> = name of the VPN  <u>vpn-id</u> = ID of the VPN  <u>extn-len</u> = length of all extensions in this VPN  <u>vpn-group</u> = optional name of the VPN group to which this VPN belongs</p>
Add a VPN group	<pre>cli vpn vpng <u>vpn-name</u> <u>vpn-group</u></pre>	<p><u>vpn-name</u> = name of the VPN  <u>vpn-group</u> = name of the VPN group to which this VPN belongs</p>
Delete a VPN	<pre>cli vpn delete <u>vpn-name</u></pre>	<u>vpn-name</u> = name of VPN to delete
List all VPNs in the database	<pre>cli vpn list</pre>	

Task	Command	Variables / Notes
Edit a VPN entry in the database	cli edit <u>vpn-name</u>	<u>vpn-name</u> = name of the VPN that you wish to edit
Initialize a database	cli db init	
Displays information about the current database	cli db info	The database name, table names and the count of records in each table are displayed.
Back up a database file to XML	cli db export <u>db-name</u>	<u>db-name</u> = name of the XML file to create and fill with database information
Create a database from exported XML	cli db create <u>db-name</u> [notadminpid]	<p><u>db-name</u> = name of the XML file from which to create a schema. The command creates all the tables using the iServer schema and imports all the data from the XML file into the new schema. The name of the new schema is derived from the XML file name, with underscores replacing all invalid characters (such as periods).</p> <p>The optional notadminpid parameter causes the pids from the XML file to be used. That is, the pid's for the imported records will not be changed to the admin pid.</p>
Add entries to a database <sup>a</sup>	cli db add <u>db-name</u>	<u>db-name</u> = name of a database file containing entries that you wish to add to the existing database
Truncates all of the specified table, except the clihistory, from the MSX database	cli db clean [all   iedge   vpn   vpng   cp   realm   igrp   vnet]	The cache for the specified table is cleared.

Task	Command	Variables / Notes
Replace entries in a database <sup>a</sup>	cli db replace <u>db-name</u>	<u>db-name</u> = name of a database file containing entries that you wish to replace in the existing database
Delete entries in a database <sup>a</sup>	cli db delete <u>db-name</u>	<u>db-name</u> = name of a database file containing entries that you wish to delete from the existing database
Drop all tables, except clihistory, from the specified schema	cli db drop <u>db-name</u>	<u>db-name</u> = name of a schema whose tables you wish to drop.
Saves a copy of the MSX schema, with data, into a new schema.	cli db save <u>db-name</u>	<u>db-name</u> = name of the new schema. New schema and tables are created and all data copied from the MSX schema to the new schema. If the specified new schema exists, or if the name contains invalid characters, an error is returned.
Print status of all configured iServer databases	cli db status	The status includes the availability and master/slave status of the database, similar to the cli rsd list command in previous CLI versions.
Switches the data of the MSX schema with data from a schema file	cli db switch <u>db-name</u>	<u>db-name</u> = name of the schema file containing the data to switch into the MSX schema. Note that the MSX cache is synchronized with the new data.
Clear <i>call limit exceeded</i> alarms	cli lsalarm clear	
Show iServer license file status	cli lstat	

Task	Command	Variables / Notes
Add a calling plan	<code>cli cp add <u>cp-name</u> [<u>route-name</u>]</code>	<u>cp-name</u> = name of the calling plan that you wish to add <u>route-name</u> = name of the call route that you wish to add under this calling plan
Delete a calling plan	<code>cli cp delete <u>cp-name</u> [<u>route-name</u>]</code>	<u>cp-name</u> = name of the calling plan that you wish to delete <u>route-name</u> = optional name of a call route to remove from this calling plan.
List all calling plans in the database	<code>cli cp list</code>	Redirect the output to a file for viewing
Edit a calling plan	<code>cli cp edit <u>cp-name</u></code>	<u>cp-name</u> = name of the calling plan that you wish to edit
Add a call route	<code>cli cr add <u>cr-name</u></code>	<u>cr-name</u> = name of the call route that you wish to add
Delete a call route	<code>cli cr delete <u>cr-name</u></code>	<u>cr-name</u> = name of the call route that you wish to delete
Edit a call route	<code>cli cr edit <u>cr-name</u> <u>attrib-name</u> <u>attrib-value</u></code>	<u>cr-name</u> = name of the call route that you wish to edit <u>attrib_name</u> is the name of an attribute to set; <u>attrib-value</u> is its setting.  Valid attributes and values list out if you enter just <code>cli cr edit</code> at the command line.
List all call routes in the database	<code>cli cr list</code>	
<b>Note:</b> This command is best avoided when working with very large databases. Output can be very large, and in most cases should be redirected to a file.		
Look up a call route by its name or phone number	<code>cli cr lkup [<u>cr-name</u>   <u>phone</u>]</code>	<u>cr-name</u> = name of the call route you wish to look up <u>phone</u> = phone number of the call route you wish to look up

Task	Command	Variables / Notes
List the status of all ongoing calls in the network	<code>cli stats</code>	See <i>Interpreting cli stats output</i> on page 114
Display statistics for SIP call processing	<code>cli stats sip</code>	Shows selected message stats for last 10 secs., 10 mins., and 10 hours., as well as current queue sizes and no. of active calls.  See <i>Interpreting cli stats output</i> on page 114
Update the iServer license file	<code>cli lupdate</code>	
Show MGK (master gatekeeper) information	<code>cli iedge cache gk</code>	
Display iServer diagnostic information	<code>cli test</code>	
Add an event trigger	<code>cli trigger add <u>new trigger name</u></code>	<u>new trigger name</u> = the name to be added (31 chars. max.)
List currently-defined triggers	<code>cli trigger list</code>	
List triggers currently in cache	<code>cli trigger cache</code>	
Remove a trigger from the cache	<code>cli trigger purge <u>trigger name</u></code>	<u>trigger name</u> = the name of the trigger to be removed
Change an event trigger	<code>cli trigger edit <u>trigger name</u> <u>parameter</u> <u>param-value</u></code>	<u>parameter</u> can be srcvendor, dstvendor, sdata, event or override. <u>param-value</u> can be req-mode-fax or h323-t38-fax
Delete an existing event trigger	<code>cli trigger delete <u>trigger name</u>&gt;</code>	<u>trigger name</u> = the name of the trigger to be deleted from the database

Task	Command	Variables / Notes
Add a new iEdge group entry	cli igrp add <u>new igrp name</u>	<u>new igrp name</u> = the name to be added (31 chars. max.)
Set parameters for an existing iEdge group	cli igrp edit <u>igrp name</u> [ <u>attrib name</u> <u>attrib value</u> ]	<u>igrp name</u> = the name of the igrp to be modified. Entering just cli igrp edit at the command line provides a list of attribute names and valid values.
Delete an existing iEdge group	cli igrp delete <u>igrp name</u>	<u>igrp name</u> = the name of the igrp to be deleted.
Show all iEdge groups currently in cache	cli igrp cache  <i>Note: In the output from this command, a zero translates to unlimited, not none.</i>	
Look up details for an existing igrp	cli igrp lkup <u>igrp name</u>	<u>igrp name</u> = the name of the igrp to be listed.  <i>Note: The information displayed by this command is taken from the cache.</i>
List all defined realms	cli realm list	
Create a new realm	cli realm add <u>new realm name</u>	<u>new realm name</u> = the name to be added (31 chars. max.)
Assign signaling Vnet(s) to a realm	cli realm edit <u>realmname</u> <u>vnetname</u> [ <u>vnet-name</u> ...]	See Chapter 24, <i>VLAN Support</i> , on page 452 for more information on vnets.
Delete an existing realm	cli realm delete <u>realm name</u>	<u>realm name</u> = the name of the realm to be deleted.
Enable one or all realms	cli realm up [ <u>realm name</u>   all]	<u>realm name</u> = the name of the realm to be enabled.
Disable one or all realms	cli realm down [ <u>realm name</u>   all]	<u>realm name</u> = the name of the realm to be disabled.

Task	Command	Variables / Notes
Enable or disable call setups on a realm	cli realm <u>realm name</u> admin [enable   disable]	<ul style="list-style-type: none"> <li>enable new call setups</li> <li>disable new call setups (doesn't end calls already in progress).</li> </ul>
Associate a realm with a far-end proxy	cli realm edit <u>realm name</u> proxy_regid <u>regid</u>	<u>realm name</u> = the name of the realm to be associated <u>regid</u> = the regid of the far-end proxy
Specify a uport on a realm's far-end proxy	cli realm edit <u>realm name</u> proxy_uport <u>uport</u>	
Configure SIP authorization for a realm	cli realm edit <u>realm name</u> sipauth [ <u>options</u> ]	See <i>Configuring Challenging (SIP), by Realm</i> on page 374 for details and available options.
Forwards keep alive invites from an application server to the UA registered through the iServer (enable), or replies with a local 200ok (disable).	cli realm edit <u>realm name</u> realm-panasonic [enable   disable]	<u>realm name</u> = the name of the realm affected by this setting.
Start or stop an iServer	iserver [ <u>daemon name</u>   all] [start   stop]	<u>daemon name</u> = name of the iServer process that you wish to start or stop
Reconfigure the iServer after making changes to the <i>servercfg</i> table	iserver [ <u>daemon name</u>   all] reconfig	<u>daemon name</u> = name of the iServer process that you wish to reconfigure
Display the status of iServer processes	iserver [ <u>daemon name</u>   all] status	<u>daemon name</u> = name of the iServer process that you wish to obtain the status for

Task	Command	Variables / Notes
Display the version of iServer software in use	iserver [ <u>daemon name</u>   all] version	<u>daemon name</u> = name of the iServer process that you wish to display the version number for
Display the time for which the iServer has been operational	iserver [ <u>daemon name</u>   all] uptime	<u>daemon name</u> = name of the iServer process that you wish to determine the uptime for

a. Note that for the **add**, **replace**, and **delete** commands, the file to be read in (i.e., the “database” named in db-name) must be properly-formatted XML. Not observing this caution may yield unpredictable results.