Cisco Prime Network Registrar IPAM 8.3 Command Line Interface (CLI) and Application Program Interface (API) Guide

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Cisco Prime Network Registrar IPAM 8.3 Command Line Interface (CLI) and Application Program Interface (API) Guide

Contents

Introduction 1

About This Guide 1

Command Line Interfaces (CLI) 2

Assumptions Regarding CLI Usage 2
Executing Commands 3
Direct 3
Indirect 3
File Format 3
Imports 6
ImportAddrpool 6
ImportAdmin 9
ImportAdminRole 15
ImportAggregateBlock 21
ImportChildBlock 23
ImportContainer 29
ImportDevice 32
ImportDeviceResourceRecord 37
ImportDhcpServer 39
ImportDNS 41
ImportDomain 48
ImportDomainResourceRecord 50
ImportElementSnapshot 52
ImportGalaxyDomain 53
ImportNetElement 55
ImportNetworkElement 57
ImportNetElementInterface 60
ImportNetService 61
ImportNetServiceWithTemplate 64
ImportNetworkLink 66
ImportPrefixPool 68
ImportRootBlock 70
ImportServiceSnapshot 72
ImportZone 73
ImportZoneResourceRecord 76
RestoreDeletedDevice 78
RestoreDeletedResourceRecords 80
Exports 82
ExportAdmin 83
ExportAdminRole 87
ExportChildBlock 90
ExportContainer 97
ExportDevice 99

ExportDeviceResourceRecord....103
ExportDeviceRestoreList....106
ExportDomainResourceRecord....109
ExportNetElement....112
ExportNetworkElement....115
ExportNetService....118
ExportNetworkLink....121
ExportPrefixPool....123
ExportResourceRecordPendingApproval....126
ExportResourceRecordPendingApprovalStatus....129
ExportResourceRecordRestoreList....131
ExportRootBlock....134
Deletes....137
DeleteAddrPool....137
DeleteAdmin....139
DeleteAdminRole....140
DeleteAggregateBlock....141
DeleteBlock....143
DeleteContainer....145
DeleteDevice....147
DeleteDomain....149
DeleteDeviceInterface....151
DeleteDeviceResourceRecord....153
DeleteDomainResourceRecord....155
DeleteNetElement....157
DeleteNetElementInterface....159
DeleteNetService....161
DeleteNetworkElement....163
DeleteNetworkLink....165
DeletePrefixPool....166
DeleteTask....168
DeleteZone....169
DeleteZoneResourceRecord....171
Gets....173
GetContainerParentHierarchy....173
GetEffectiveDhcpServersForContainer....174
Tasks....176
ArpDiscoverNetElement Task....176
DhcpConfigurationTask....178
DHCPUtilization....180
DiscoverNetElement....181
DnsConfigurationTask....183
GlobalRollup....185
GlobalSync....186
TaskStatus....187
Updates....189
DetachBlock....189
DetachContainer....191
JoinBlock....193
ModifyAddrpool....194
ModifyBlock....197
ModifyContainer....203

ModifyDevice 208
ModifyDeviceResourceRecord 212
ModifyDhcpServer 215
ModifyDomainResourceRecord 218
ModifyNetElementInterface 221
ModifyPendingApproval 223
ModifyPrefixPool 225
SplitBlock 228
UseNextReservedIPAddress 230
Utilities 231
DhcpRelease 231
Purge 234

Application Program Interfaces (API) 237

Using the API 237
Invoking the web service and authentication 237
Error Processing 239
Available Application Program Interface Matrix 239
Imports 242
AddSite 242
DetachBlock 246
ImportAddressPool 248
ImportAdmin 252
ImportAdminRole 255
ImportAggregateBlock 263
ImportChildBlock 266
ImportContainer 272
ImportDevice 277
ImportDeviceResourceRecord 282
ImportDhcpServer 284
ImportDomain 288
ImportDomainResourceRecord 291
ImportGalaxyDomain 293
ImportNetElement 295
ImportNetworkElement 298
ImportNetElementInterface 303
ImportNetService 305
ImportNetworkLink 308
ImportPrefixPool 311
ImportRootBlock 314
ImportZone 317
JoinBlock 321
ModifyBlock 322
ModifyPendingApproval 330
RestoreDeletedDevice 332
RestoreDeletedResourceRecord 334
SplitBlock 336
Gets 338
getAddressPool 338
getAdmin 339
getAdminRole 340
getBlock 341
getContainer 343

getContainerParentHierarchy....344
getDevice....345
getDeviceResourceRec....347
getDhcpServer....349
getDomainResourceRec....350
getEffectiveDhcpServersForContainer....351
getNetworkElement....352
getNetelementInterface....354
getNetworkLink....355
getPrefixPool....356
Tasks....357
ArpDiscoverNetElement....357
DHCPConfigurationAllFiles....358
DHCPUtilization....359
DiscoverNetElement....361
DNSConfigurationAllFiles....362
DNSConfigurationChangedZones....364
DNSConfigurationSelectedZones....365
DNSDDNSAllRRs....367
DNSDDNSChangedRRs....368
GetTask....369
GetTaskStatus....370
GlobalNetElementSync....371
GlobalNetServiceSync....372
GlobalRollup....373
Exports....374
Overview....374
Export Categories....374
Legacy Web Services....374
Next Generation Web Services....375
Legacy Web Services....376
ExportNetElementsAsCSV....376
ExportAllNetElementsAsCSV....379
ExportNetServicesAsCSV....380
ExportAllNetServicesAsCSV....383
Next Generation Web Services....384
Selectors....384
Options....384
Paging....384
Sessions....385
ExportAdmin....386
ExportAdminRole....388
ExportRootBlock....390
ExportChildBlock....393
ExportContainer....399
ExportDevice....401
ExportDeviceResourceRecord....407
ExportDeviceRestoreList....411
ExportDomainResourceRecord....415
ExportNetworkElement....419
ExportNetworkLink....423
ExportPrefixPool....426

ExportResourceRecordPendingApproval 429
ExportResourceRecordPendingApprovalStatus 432
ExportResourceRecordRestoreList 435
Updates 439
UseNextReservedIPAddress 439
Deletes 441
DeleteAddrPool 441
DeleteAdmin 443
DeleteAdminRole 445
DeleteAggregateBlock 446
DeleteBlock 448
DeleteContainer 450
DeleteDevice 451
DeleteDeviceInterface 453
DeleteDeviceResourceRecord 456
DeleteDomain 459
DeleteDomainResourceRecord 461
DeleteNetElement 463
DeleteNetElementInterface 465
DeleteNetService 467
DeleteNetworkElement 469
DeleteNetworkLink 471
DeletePrefixPool 473
DeleteTaskByDate 475
DeleteTaskByDays 476
DeleteTaskById 477
DeleteZone 478
DeleteZoneResourceRecord 480
DetachContainer 482

Other Interfaces 484

Callout Manager 484
Operation 484
Configuration 484
RIR Template Support 490
Introduction 490
Configuration 490
Operation 492
RIR REST Interface Support 495
Introduction 495
Description 495
Configuration 495
Notes about Block Modification 505
DNS Listener 508
Configuration 508
Record Processing Rules 511
Detailed Description 512
DNS Deployment Callout 514

Appendix A API Changes 515

IPAM 8.3 515
IPAM 8.1.3 517
IPAM 8.1.2 520

Appendix B – RIR Callout 524

Sample rir_callout.ini File 524

Appendix C REST API 530

Using the API 530

Swagger Interface and Tools 531

Authentication 531

Introduction

About This Guide

This guide outlines command line interfaces (CLIs) into Cisco Prime Network Registrar IPAM 8.3 and application programming interfaces (APIs) to Cisco Prime Network Registrar IPAM.

Using CLIs extends the effectiveness of the IPAM Administrator, allowing him or her flexibility to run IPAM functions from a command line. Often this can shorten the time needed to bulk import or export data, or can allow for scheduling of tasks outside the IPAM product using *cron* or Windows Task Scheduler.

Using APIs extends the effectiveness of the IPAM Administrator, allowing him or her flexibility to programmatically interface to IPAM. This enables the integration of IPAM into business processes or custom workflow.

Note: IPAM 8.3 and later versions will not support Solaris. Refer to earlier versions of IPAM documents if you want to use IPAM with Solaris support.

Command Line Interfaces (CLI)

Assumptions Regarding CLI Usage

Each CLI performs a specific task, or in some cases, several tasks at once. However, there are assumed dependencies among the different CLIs such that some CLIs will not function properly unless either other CLIs are run or some manual data setup is performed.

The following manual data setup is recommended to populate the initial IPAM database before running any CLIs:

- Manual step – create block types
- Manual step – create device types
- Manual step – create user defined fields
- Manual step – create IP allocation reasons
- Manual step - create IP address allocation templates
- Manual step - create DNS and/or DHCP servers

Table 1 illustrates the recommended order in which the IPAM CLIs should be run.

Table 1 CLI Sequence

Sequence	CLI Name	Brief Description	Data Dependencies
1	ImportContainer (logical)	Imports logical containers	N/A
2	ImportNetElement	Imports network elements such as routers and switches	N/A
3	ImportContainer (device)	Imports device containers	Network elements created using **`ImportNetElement`** CLI or created manually
4	ImportRootBlock	Imports root IP address blocks	Containers created using **`ImportContainer`** CLI or created manually
5	ImportChildBlock	Imports child IP address blocks	Root blocks created using **`ImportRootBlock`** CLI or created manually
6	ImportDevice	Imports devices	Blocks created using **`ImportChildBlock`**, **`ImportRootBlock`**, or created manually

Sequence	CLI Name	Brief Description	Data Dependencies
7	DiscoverNetElement	Discovers live network element data	Network elements created using **`ImportNetElement`** CLI or created manually
8	DHCPUtilization	Discovers live DHCP utilization data	DHCP servers created manually
9	ImportElementSnapshot	Imports network element data	Data generated from **`DiscoverNetElement`** CLI
10	ImportServiceSnapshot	Imports address pools discovered by "Collect DHCP Utilization" or "Global Synchronization of DHCP Servers" tasks.	Data generated from **`DHCPUtilization`** CLI
11	ImportDNS	Imports DNS domain and zone data	DNS servers and views created manually

Executing Commands

Each CLI is capable of being executed either directly by invoking the Java JVM, or indirectly via the available command script. The direct approach requires a rather lengthy and cumbersome syntax, while the indirect method requires the proper passing of necessary parameters.

Direct

The following is an example of the direct method of execution (it assumes that the IPAM environment variables, namely **`INC_HOME`**, **`JAVA_HOME`** and **`CLASSPATH`** are resident):

```
$INCHOME/jre/bin/java -DINC_HOME=$INCHOME -DNCX_HOME=$NCX_HOME -Duser.dir=$INCHOME
-cp $CLASSPATH com.diamondip.netcontrol.cli.ImportNetService -u joe -p joepwd
-f southeast.csv
```

Indirect

The following example executes the same call but uses the indirect approach of calling a predefined command script:

```
/opt/incontrol/ImportNetService.sh -u joe -p joepwd -f southeast.csv
```

File Format

The format for import files is comma-separated values (CSV) and Microsoft Excel Workbook (.xls or .xlsx). These files are easily created or modified using any standard text editor. For greater ease of use, most spreadsheet applications like Microsoft Excel or OpenOffice Calc support saving as a CSV format.

Template files for each CLI are available in the *templates* directory underneath the CLI directory (typically *<product home>/etc/cli*). These include a comment line, beginning with "#", that provides a label for each column.

Note when creating import files, any lines that begin with the pound (#) character are ignored by the InControl CLIs.

Available Command Line Interface Matrix

Object	Import	Modify	Delete	Export
Address Pool	X	X	X	
Administrator	X	*(see ImportAdmin)*	X	X
Administrator Role	X	*(See ImportAdmin Role)*	X	X
Aggregate Block	X		X	
Child Block	X	X	X	X
Container	X	X	X	X
Device	X	X	X	X
Device Interface			X	
Device RR	X	X	X	X
DHCP Server	X	X		
DNS	X			
Domain	X		X	
Domain RR	X	X	X	X
Galaxy Domain	X			
Network Element	X	*(see ImportNetwork Element)*	X	X
Net Element Interface	X	X	X	
Net Service	*(use ImportDhcp Server)*		X	X
Network Link	X	*(see ImportNetwork Link)*	X	X
Prefix Pool	X	X	X	X
RR Pending Approval		X		X
RR Pending Approval Status				X
Root Block	X	X	X	X
Zone	X	*(see ImportZone)*	X	
Zone RR	X		X	
Next Available IP	X			
Join Block		X		
Split Block		X		
Detach Block		X		

Task	Import	Modify	Delete	Export
GlobalNetElementSync	X		X	
GlobalNetServiceSync	X		X	
ImportElementSnapshot	X			

ImportServiceSnapshot	X			
GlobalRollup	X		X	
DiscoverNetElement	X		X	
DhcpConfigurationTask	X			
DhcpUtilization	X		X	
GetTask	X			
GetTaskStatus	X			
DeleteTask			X	

Imports

ImportAddrpool

Overview

The **`ImportAddrpool`** CLI allows the user to bulk import address pools into IPAM.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.ImportAddrpoolCLI -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportAddrpool.sh -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportAddrpool.cmd -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the file to import. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-imported) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example imports address pools from the *newaddrpools.csv* file, places into the *newaddrpools.reject* file any records that could not be imported, and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/ImportAddrpool.sh -u joe -p joepwd -f newaddrpools.csv
-r newaddrpools.reject -e importerrors.txt
```

File Format

Col	Field	Accepted Values	Required
A	Start Address	The IP Address of the first address in the pool. This address must be in a block with an In-Use/Deployed status.	Yes

Col	Field	Accepted Values	Required
B	End Address	The IP Address of the last address in the pool. This address must be in the same block as the Start Address. In addition, the Start and End addresses must not overlap any other pools. This is ignored for IPv6 pools.(PrefixLength is used instead).	Yes, for IPv4 pool.
C	Address Pool Type	One of "Dynamic DHCP", "Automatic DHCP", "Static", "Reserved", "Dynamic NA DHCPv6", "Automatic NA DHCPv6", "Dynamic TA DHCPv6", "Automatic TA DHCPv6"	Yes
D	Name	Address Pool name. Defaults to "Start Address-End Address"	No
E	Share Name	DEPRECATED. This field will be ignored.	No
F	Container	The name of the container that holds the block in which the pool is defined. This is required only if there is overlapping address space in use, and the start address is in overlapping space. The container is then used to uniquely determine the block that will contain the address pool.	No, unless Start address is not unique.
G	Primary Net Service	The name of the DHCP server that will serve addresses from this pool. Note: This is required when a DHCP server is not defined for the subnet, and this address pool is one of the DHCP address types: "Dynamic DHCP", "Automatic DHCP", "Dynamic NA DHCPv6", "Automatic NA DHCPv6", "Dynamic TA DHCPv6", "Automatic TA DHCPv6"	See note
H	Failover Net Service	The name of the failover DHCP server that will serve addresses from this pool. To use this field, Primary Net Service must also be specified.	No
I	DHCP Option Set	The name of an Option Set used with this pool. For IPV4 address pools, the DHCP option set applies only to non-CNR DHCP servers. For IPV6 address pools, the DHCP option set applies only to CNR DHCP servers.	No
J	DHCP Policy Set	The name of a Policy Set used with this pool. For IPV4 address pools, the DHCP policy set applies only to non-CNR DHCP servers. For IPV6 address pools, the DHCP policy set applies only to CNR DHCP servers.	No
K	Allow DHCP Client Classes	For IPv4 and IPv6 Dynamic and Automatic type pools: A list of Client Classes that are allowed in this address pool. Separate the list entries with a vertical bar "|". For example, to allow two client classes named "allowA" and "allowB", specify: allowA|allowB	No
L	Deny DHCP Client Classes	For IPv4 and IPv6 Dynamic and Automatic type pools: A list of Client Classes that are NOT allowed in this address pools. Separate the list entries with a vertical bar "|". For example, to disallow two client classes named "denyA" and "denyB", specify: denyA|denyB	No

ImportAddrpool

Col	Field	Accepted Values	Required
M	PrefixLength	CIDR size of the pool for an IPv6 pool. An IPv6 should be on CIDR boundaries.This is ignored for an IPv4 pool(End Address field is used instead).	Yes, for IPv6 pool.
N	OverlapInterfaceIp	Flag to allow a DHCPv6 pool to overlap an interface address. This flag may be set only if pool is managed by a CNR DHCPv6 server. This is ignored for DHCPv4 pools.	No

ImportAdmin

Overview

The **ImportAdmin** CLI allows the user to bulk import administrators into IPAM.

Note that while administrators of administrator type "NORMAL" can import and modify administrators and their roles, only MASTER administrators can import or update administrator policies and assignable roles (columns O-Y, see below). NORMAL administrators will receive an error message if policies or assignable roles are specified on import. On an import request to modify an administrator, if a NORMAL administrator specifies policies or assignable roles, that information is ignored.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.ImportAdminCLI -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-o] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportAdmin.sh -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-o] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportAdmin.cmd -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-o] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the file to import. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-imported) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.
-o	No	Overwrite option. Specify whether the import file contents will overwrite any matching records that exist in the database. See below.
-v	No	Produces verbose output.

Usage Example

This example imports administrators from the *newadmins.csv* file, places into the *newadmins.reject* file any records that could not be imported, and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/ImportAdmin.sh -u joe -p joepwd -f newadmins.csv
-r newadmins.reject -e importerrors.txt
```

Import with overwrite

This example imports administrators with the overwrite option:

```
$INCHOME/etc/cli/ImportAdmin.sh –u joe –p joepwd –f admins.csv –o
```

Note the following:

- You can produce a file in the required format using the ExportAdmin CLI.
- The –o (overwrite) parameter is required in order to modify administrators via this CLI.
- The rows describe administrators to be either imported or modified. In other words, existing administrators will be modified; new administrators will be imported. Administrators to be modified are identified by administrator login id.
- The login id cannot be modified.
- An empty cell indicates no change will be made for that field.
- To clear a field in an existing record, specify "!BLANK!". This is valid for columns C-K, O-P.

File Format

Col	Field	Accepted Values	Required
A	Login ID	Administrator login ID (for example, jsmith10)	Yes
B	Login Password	Administrator's password. (Import only; cannot be overwritten)	Yes
C	First Name	First (given) name of administrator	No
D	Last Name	Last (family) name of administrator	No
E	Address1	Mailing address	No
F	Address2	Additional line for mailing address	No
G	Address3	Additional line for mailing address	No
H	Email	Email address	No
I	Phone	Phone number	No
J	Pager	Paging number	No
K	Fax	Fax number	No
L	Authorize Externally	Determines whether this user's username and password will be passed to an external authentication system. This is setup by the IPAM administrator in the System Policies screen. Accepted values are **true** or **false**.	No, defaults to **false**
M	Enabled	Determines whether this user can access the IPAM system. Accepted values are **true** or **false**.	No, defaults to **true**

Col	Field	Accepted Values	Required
N	Admin Type	Specify one of **MASTER**, **NORMAL**, or **READONLY**. **MASTER** users have full control over the entire IPAM system. **NORMAL** users have ordinary read-write permissions, and may be restricted to working with only certain portions of the IPAM system. **READONLY** users have read-only access to IPAM, and may be restricted to seeing only certain portions of the IPAM system.	No, defaults to **READONLY**
O	Role	Administrator Roles for this administrator. Specify multiple roles by separating with '|'.	No
P	Assignable Role	Assignable Administrator Roles for this administrator. Assignable Roles are roles that the Administrator can assign to another Administrator. Specify multiple roles by separating with '|'.	No
Q	Authorized Functions	**ALL**, **NONE**, or a list of functions and main headings the administrator can access, separated by '|'. Each function corresponds to a menu item that this administrator is allowed to access. List can include a main heading like **IPAM**. List can include individual functions like **AddSites**. A complete list of keywords for the functions and headings is provided below. These keywords are case-insensitive.	No; defaults to **NONE**
R	Container Access Control	**ALL**, **NONE**, or a list of containers with access indication. Separate multiple containers using '|'. Separate access indicators using '^'. The access indicators are specified as: Name^read^write^delete^apply to children^ *device approve access For example: InControl/US/ East^Y^N^N^Y|West^Y^Y^Y^Y **Only when Workflow Type system policy is set to "Device"* Note: If you specify **NONE**, IPAM expects that there are no Block Access Control rules.	No; defaults to **NONE**

Col	Field	Accepted Values	Required
S	Block Access Control	**NONE** or a list of blocks with access indication. This is used to fine-tune the blocks given access by the container access control rules. You typically add blocks to an administrator role ACL when you want to override the privileges defined on the container for a specific block. Separate multiple blocks using '|'. Separate access indicators using '^'. The access indicators are specified as: Block name^container name^block status^ read^write^delete^ *device approve access For example: 10.0.0.0/24^InControl/US/East^^Y^N^N^Y | 2001::4:0/110^West^Aggregate^Y^Y^Y^Y **Only when Workflow Type system policy is set to "Device"* Block status is optional. When multiple blocks with the same name are found in a container, specify block status to resolve the conflict. This typically occurs when an aggregate block and another block have the same block name. Note: IPAM expects that the container specified in this rule has a Container Access Control rule defined.	No; defaults to reflect the access defined by the container access control rules
T	Block Type Access	**ALL**, **NONE**, or a list of allowed block types. Only required if there are limitations. For example: Any|Data|VoIP To limit allowed block sizes, specify the allowed sizes, or **NONE**, following the block type, separated by '/'. Any/*V4 specification*/*V6 specification* | Data/*V4 specification*/*V6 specification* where *V4* and *V6 specification* are a list of ^ separated block sizes: Any/24^25|Data/22^23/65^66|VoIP//67^68 The default, with no block sizes specified, as for Any V6 and VoIP V4 in the above example, indicates all sizes are allowed, which sets the same option as the GUI's: "Abstain and leave block sizes 'undecided'". Any/24^25|Data/22^23/65^66|VoIP/NONE/67^68 In the above example, VoIP is limiting block sizes to allow only 2 V6 block sizes.	No; defaults to **ALL** No; defaults to all sizes allowed

Col	Field	Accepted Values	Required
U	Device Type Access	**ALL**, **NONE**, or a list of allowed device types, as configured in IPAM. Only required if there are limitations. For example: PC|Server|Switch	No; defaults to **ALL**
V	Policies	Policy selections are specified in the following order, separated by '/': CLI Access/duplicate hostname checking/checking style/allow dup A recs/allow dup hardware addr CLI Access: **true** or **false** Dup hostname checking: **I**(gnore), **W**(arn) or **F**(ail) Dup hostname checking style: **F** (fully qualified) or **H** (hostname only) Allow duplicate A records: **I**(gnore), **W**(arn) or **F**(ail) Allow duplicate hardware addresses: **I**(gnore), **W**(arn) or **F**(ail) For example: false/I/H/W/F	No; defaults to: false/W/F/W/W
W	Domain Access Control	**ALL**, **NONE**, or a list of domains with access indication. Separate multiple domains with '|'. Separate access indicators using '^'. The access indicators are specified as: Domain name/domain type ^read^write^delete^RR access^RR write^apply to children^*Resource Record Approve Access For example: sample.com/default^Y^N^N^N^N^Y **Only when Workflow Type system policy is set to "Resource Records"*	No; defaults to **NONE**
X	Net Service Access Control	**ALL**, **NONE**, or a list of servers with access indication. Separate multiple services with '|'. Separate access indicators using '^'. The access indicators are specified as: Name/type^read^write^deploy Specify type as either **DNS** or **DHCP**. For example: DnsOne/DNS^Y^N^N	No; defaults to **ALL**
Y	Resource Record Type Access Control	**ALL**, **NONE**, or a list of allowed resource record types. Only required if there are limitations. For example: A|PTR|CNAME *Note: "NSAP-PTR" is specified as* ***NSAPPTR*** *and "Zone RR" is specified as* ***ZONERR****.*	No; defaults to **ALL**
Z	Address Type Access	**ALL**, **NONE**, or a list of allowed address types. Only required if there are limitations. For example: STATIC|RESERVED	No; defaults to **ALL**

Authorized Functions (column Q)

The authorized functions have two formats that can be used together: main headings and individual functions. Their specification is case-insensitive. You can find the complete list in the ImportAdminRole description.

Specifying a heading means all the functions for that heading are authorized.

Specifying a function means it is authorized. If a function is not listed, it is not authorized, unless its header was specified.

In following example, the only functions allowed for this administrator are all functions under the IPAM and DHCP headings, as well as the individual functions "Tasks" and "Alerts":

```
IPAM|Tasks|Alerts|DHCP
```

ImportAdminRole

Overview

The **ImportAdminRole** CLI allows the user to bulk import administrator roles into IPAM. Note that only administrators of administrator type "MASTER" can invoke this CLI.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.ImportAdminRoleCLI -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-o] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportAdminRole.sh -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-o] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportAdminRole.cmd -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-o] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the file to import. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-imported) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.
-o	No	Overwrite option. Specify whether the import file contents will overwrite any matching records that exist in the database. See below.
-v	No	Produces verbose output.

Usage Example

This example imports administrator roles from the *newadminroles.csv* file, places into the *newadminroles.reject* file any records that could not be imported, and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/ImportAdminrole.sh -u joe -p joepwd -f newadminroles.csv
-r newadminroles.reject -e importerrors.txt
```

Import with overwrite

This example imports administrator roles with the overwrite option:

```
$INCHOME/etc/cli/ImportAdminRole.sh –u joe –p joepwd –f adminroles.csv –o
```

Note the following:

- You can produce a file in the required format using the ExportAdminRole CLI.
- The –o (overwrite) parameter is required in order to modify administrator roles via this CLI.
- The rows describe administrator roles to be either imported or modified. In other words, existing administrator roles will be modified; new administrator roles will be imported. Administrator roles to be modified are identified by role name.
- The role name cannot be modified.
- An empty cell indicates no change will be made for that field.
- To clear the description, column B, in an existing record, specify "!BLANK!".

File Format

Col	Field	Accepted Values	Required
A	Role Name	Specify the name of the administrator role.	Yes
B	Description	Specify a description of this administrator role.	No
C	Authorized Functions	**ALL**, **NONE**, or a list of functions and main headings the administrator can access, separated by '|'. Each function corresponds to a menu item that this administrator is allowed to access. List can include a main heading like **IPAM**. List can include individual functions like **AddSites**. A complete list of keywords for the functions and headings is provided below. These keywords are case-insensitive.	No; defaults to **ALL**
D	Container Access Control	**ALL**, **NONE**, or a list of containers with access indication. Separate multiple containers using '|'. Separate access indicators using '^'. The access indicators are specified as: Name^read^write^delete^apply to children^ *device approve access For example: InControl/US/ East^Y^N^N^Y|West^Y^Y^Y^Y **Only when Workflow Type system policy is set to "Device"* Note: If you specify **NONE**, IPAM expects that there are no Block Access Control rules.	No; defaults to **ALL**: InControl^Y ^Y^Y^Y^*N* **and no device approval access*

Col	Field	Accepted Values	Required
E	Block Access Control	**NONE** or a list of blocks with access indication. This is used to fine-tune the blocks given access by the container access control rules. You typically add blocks to an administrator role ACL when you want to override the privileges defined on the container for a specific block. Separate multiple blocks using '|'. Separate access indicators using '^'. The access indicators are specified as: Block name^container name^block status^ read^write^delete^ *device approve access For example: 10.0.0.0/24^InControl/US/East^^Y^N^N^Y | 2001::4:0/110^West^Aggregate^Y^Y^Y^Y **Only when Workflow Type system policy is set to "Device"* Block status is optional. When multiple blocks with the same name are found in a container, specify block status to resolve the conflict. This typically occurs when an aggregate block and another block have the same block name. Note: IPAM expects that the container specified in this rule has a Container Access Control rule defined.	No; defaults to reflect the access defined by the container access control rules
F	Block Type Access	**ALL**, **NONE**, or a list of allowed block types. Only required if there are limitations. For example: Any|Data|VoIP To limit allowed block sizes, specify the allowed sizes, or **NONE**, following the block type, separated by '/'. Any/*V4 specification*/*V6 specification* | Data/*V4 specification*/*V6 specification* where *V4* and *V6 specification* are a list of ^ separated block sizes: Any/24^25|Data/22^23/65^66|VoIP//67^68 The default, with no block sizes specified, as for Any V6 and VoIP V4 in the above example, indicates all sizes are allowed, which sets the same option as the GUI's: "Abstain and leave block sizes 'undecided'". Any/24^25|Data/22^23/65^66|VoIP/NONE/67 ^68 In the above example, VoIP is limiting block sizes to allow only 2 V6 block sizes.	No; defaults to **ALL**
G	Device Type Access	**ALL**, **NONE**, or a list of allowed device types, as configured in IPAM. Only required if there are limitations. For example: PC|Server|Switch	No; defaults to **ALL**

Col	Field	Accepted Values	Required
H	Policies	Policy selections are specified in the following order, separated by '/': CLI Access/duplicate hostname checking/checking style/allow dup A recs/allow dup hardware addr CLI Access: **true** or **false** Dup hostname checking: **I**(gnore), **W**(arn) or **F**(ail) Dup hostname checking style: **F** (fully qualified) or **H** (hostname only) Allow duplicate A records: **I**(gnore), **W**(arn) or **F**(ail) Allow duplicate hardware addresses: **I**(gnore), **W**(arn) or **F**(ail) For example: false/I/H/W/F	No; defaults to: false/W/F/W/W
I	Domain Access Control	**ALL**, **NONE**, or a list of domains with access indication. Separate multiple domains with '|'. Separate access indicators using '^'. The access indicators are specified as: Domain name/domain type ^read^write^delete^RR access^RR write^apply to children^*Resource Record Approve Access For example: sample.com/default^Y^N^N^N^N^Y **Only when Workflow Type system policy is set to "Resource Records"*	No; defaults to **ALL**. A rule for each domain type is defined as: *./type*^Y^Y^Y^Y^Y^Y^Y^Y **including device approval access*
J	Net Service Access Control	**ALL**, **NONE**, or a list of servers with access indication. Separate multiple services with '|'. Separate access indicators using '^'. The access indicators are specified as: Name/type^read^write^deploy Specify type as either **DNS** or **DHCP**. For example: DnsOne/DNS^Y^N^N	No; defaults to **ALL**
K	Resource Record Type Access Control	**ALL**, **NONE**, or a list of allowed resource record types. Only required if there are limitations. For example: A|PTR|CNAME *Note: "NSAP-PTR" is specified as* ***NSAPPTR*** *and "Zone RR" is specified as* ***ZONERR****.*	No; defaults to **ALL**
L	Address Type Access	**ALL**, **NONE**, or a list of allowed address types. Only required if there are limitations. For example: STATIC|RESERVED	No; defaults to **ALL**

Authorized Functions (column C)

The authorized functions have two formats that can be used together: main headings and individual functions. Their specification is case-insensitive. You can find the list, below.

Specifying a heading means all the functions for that heading are authorized.

Specifying a function means it is authorized. If a function is not listed, it is not authorized, unless its header was specified.

In following example, the only functions allowed for this administrator are all functions under the IPAM and DHCP headings, as well as the individual functions "Tasks" and "Alerts":

```
IPAM|Tasks|Alerts|DHCP
```

Authorized Functions

The following are the main headings, corresponding to the headings on the Authorized Functions tab:

IPAM, DNS, DHCP, UTILIZATION, AUDIT, REPORTSOTHER, SYSTEM, SUBNETBLOCK, IPDEVICES, TOOLSOTHER, APPLIANCES, ADMINISTRATORS

The following are the individual functions, displayed by their main headings:

IPAM:

ContainerView	AddSites	SubnetBlockView	SubnetPolicy
VlanPolicy	IpDomainSearch	IPResourceRecTab	GeneralTab
AddBlock	DeleteBlock	SplitBlock	JoinBlock
MoveBlock	AttachBlocks	DetachBlocks	Discovery
AddressSpaceReclaim	ContainerMaintenance	NetworkElementsDevices	ServerPairs
PendingApprovals			

DNS:

DNSServersServices	ExpertOptions	DNSConfigDeploy	DNSAllFiles
ChangedZones	SelectedZones	ConfigOnly	ChangedRRsViaDDNS
AllRRsViaDDNS	AllUserCreatedRRsViaDDNS	Domains	Galaxies
LogChannels	ServerTemplates	DomainTypes	AddressMatchLists
UpdatePolicies	TransactionKeys	DNSOptionVendorDictionary	
DNSOptionMasterDictionary	DNSSoftwareProducts		

DHCP:

DHCPServersServices	UtilizationView	NetworkLinks	DHCPConfigDeploy
DHCPAllFiles	PolicySets	OptionSets	ClientClasses
DHCPOptionVendorDictionary	DHCP OptionMasterDictionary	DHCPSoftwareProducts	

UTILIZATION:

ContainerUtilization	SubnetBlockUtilization	LowPoolReport

AUDIT:

ContainerAudit	SubnetBlockAudit	IPDeviceAudit	ResourceRecAudit	AdminActivityAudit
AdminLoginAudit	DelegatedPrefixAudit			

REPORTSOTHER:

Tasks	Alerts	ApplianceDashboard	LoggedInAdmins	RIRSummary	SWIPNetNameReport

SYSTEM:

PoliciesAndOptions	Agents	ImportWizard	Search

SUBNETBLOCK:
AllocationReasonCodes BlockTypes AddressPoolAllocTemplates SiteAllocTemplates RIROrganizationIDs

IPDEVICES:
VendorModels DeviceTypes NamingPolicies InterfaceTemplates

TOOLSOTHER:
ThresholdSets BlockThresholdAlerts ContainerThresholdAlerts NetworkServicesThresholdAlerts
UserDefinedFields InformationTemplate

APPLIANCES:
ApplianceDefinition ManageAppliance SoftwareUpdates SoftwarePackages

ADMINISTRATORS: AdministratorDefinition AdministratorRoles
Misc (no heading): FileChangePassword

ImportAggregateBlock

Overview

The **ImportAggregateBlock** CLI allows the user to insert an intermediate level Aggregate block between existing blocks in the block hierarchy. By specifying a parent block, target block and a container, IPAM will validate and insert the desired aggregate block. It will also adjust the parent block assignments of any would-be child blocks.

Usage

Direct

```
$INCHOME/jre/bin/java –cp $CLASSPATH
com.diamondip.netcontrol.cli.ImportAggregateBlockCLI –u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportAggregateBlock.sh –u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportAggregateBlock.cmd –u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the file to import. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-imported) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example imports aggregate blocks from the *newaggblocks.csv* file, places into the *newaggblocks.reject* file any records that could not be imported, and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/ImportAggregateBlock.sh –u joe –p joepwd –f newaggblocks.csv
–r newaggblocks.reject –e importerrors.txt
```

File Format

Col	Field	Accepted Values	Required
A	Container	The name of the container into which to insert the new aggregate block. Names can be in either short or long format. Short format example: Dallas. Long format example: InControl/Texas/Dallas. Long format eliminates ambiguity in cases where there are duplicate container names.	Yes

ImportAggregateBlock

Col	Field	Accepted Values	Required
B	Start Address	The start address of the new aggregate block.	Yes
C	Block Size	The size of the block in short-notation (e.g., 24 for a 255.255.255.0 network).	Yes
D	Block Type	The Block Type for the block If not specified, a block type of Any is assumed.	No
E	Block Name	A name for the block. Defaults to system supplied name of Address space/Block size.	No
F	Description	A description of the block.	No
G	SWIP Name	SWIP name for the block.	Yes, if required by container rules
H	Allocation Reason	The name of a pre-existing Allocation Reason.	No
I	Allocation Reason Description	A description of the reason for the allocation. Wrap the statement in "quotes" if it contains any commas.	No
J	Interface Name	If this block is being added to a device container, the name of the interface to attach the block to.	Yes, if block is being added to device container. Otherwise, no.
K	Interface Offset or Address	DEPRECATED. This field will be ignored.	No
L	Create Reverse Domains	Whether or not to automatically create reverse DNS domain(s) for this block. Accepted values are true or false. If not specified, defaults to false.	No
M	Domain Type	Specify the domain type for the reverse DNS domain(s) to be created when Create Reverse Domains is "true". If not specified, defaults to "Default".	No
N	User Defined Fields	A series of name=value pairs, where the name is the UDF name and the value is desired value. Multiple pairs can be specified by separating each pair with the "|" character. For example, UDFone=value one |UDFtwo=value two. If the UDF type is Checkbox, the valid values are "on" or "off". If the UDF type is Textarea, use "\n" to separate lines.	Yes, for UDFs defined as required fields.
O	Parent Container	The name of the container where the parent block resides.	Yes
P	Parent Block Address	The address of the parent block	Yes
Q	Parent Block Size	The size of the parent block in short-notation (e.g., 24 for a 255.255.255.0 network).	Yes

ImportChildBlock

Overview

The **ImportChildBlock** CLI allows the user to bulk import child blocks into IPAM. It also allows modification of existing records when using the overwrite option (-o) and the expanded format, described later in this section. It can also be used to attach an existing block to another container, by specifying an existing Address Block.

Usage

Direct

```
$INCHOME/jre/bin/java –cp $CLASSPATH
com.diamondip.netcontrol.cli.ImportChildBlockCLI –u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-o] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportChildBlock.sh –u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-o] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportChildBlock.cmd –u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-o] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the file to import. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-imported) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.
-o	No	Overwrite option. Specify whether the import file contents will overwrite any matching records that exist in the database. This option requires expanded format. See below.
-v	No	Produces verbose output.

Usage Example

This example imports child blocks from the *newchildblocks.csv* file, places into the *newchildblocks.reject* file any records that could not be imported, and reports errors to the *importerrors.txt* file.

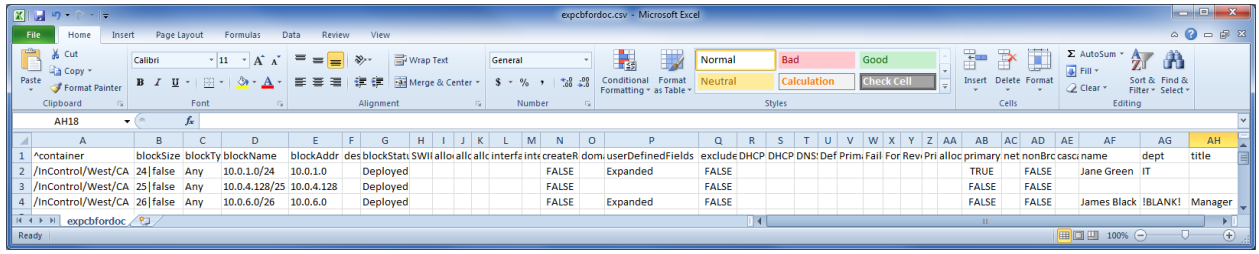
```
$INCHOME/etc/cli/ImportChildBlock.sh –u joe –p joepwd –f newchildblocks.csv
–r newchildblocks.reject –e importerrors.txt
```

Import with overwrite using expanded format and the !BLANK! keyword

This example imports child blocks using the expanded format.

```
$INCHOME/etc/cli/ImportChildBlock.sh -u joe -p joepwd -f cexport.csv -o
```

Here is the input file:



Note the following:

- You can produce a file in this format using the ExportChildBlock CLI.
- The first row is a header line, beginning with "^". The column headers beyond the architected column AE contain the user defined field tags. This row is required in order to modify blocks via this CLI, regardless of whether or not the expanded format is being used.
- The –o (overwite) parameter is required in order to modify blocks via this CLI.
- The rows describe blocks to be either imported or modified. In other words, existing blocks will be modified; new blocks will be imported. Blocks to be modified are identified by either blockName (D) or blockAddr (E) and, when necessary to resolve ambiguity, container (A).
- Column P contains "Expanded" when the row includes user defined fields beyond column AE. For example, row 3 has no user defined fields defined for that block.
- In this example, columns AF-AH contain the values for the user defined fields. To clear a field in an existing record, specify "!BLANK!", as shown in row 4 column AG. A cell with no value, as in row 2 column AH, will result in no change to the value for the record stored in the database.
- Blocks cannot be moved to a different container using this CLI, because the container is used to identify the block.
- If an expanded user defined field column contains data for a block where that field is not defined, an error will be reported.

File Format

Col	Field	Accepted Values	Required
A	Container	The name of the container that will hold the block. Names can be in either short or long format. Short format example: **Dallas**. Long format example: **InControl/Texas/Dallas**. Long format eliminates ambiguity in cases where there are duplicate container names.	Yes
B	Block size | IPV6	The size of the block in short-notation (e.g., 24 for a 255.255.255.0 network). If an IPV6 block is desired, follow the block size with "|true". IPV4 is the default.	Yes | No
C	Block type	The Block Type for the block If not specified, a block type of **Any** is assumed.	No
D	Block Name	A name for the block. Defaults to system supplied name of *Address space/Block size*.	No
E	Address Block	The address block to allocate. If no address block is specified, space will be auto-allocated. If the address is specified and already exists, the block will be attached to the specified container.	No
F	Description	A description of the block. Use "\n" to separate lines.	No
G	Current Status	The current status of the block. Accepted values are: **Deployed**, **FullyAssigned**, **Reserved**, **Aggregate**.	Yes
H	SWIP Name	SWIP name for the block.	Yes, if required by Container rules
I	Allocation Reason	The name of a pre-existing Allocation Reason. If Allocation Reason is not currently in IPAM, this field is skipped.	No
J	Allocation Reason Description	A description of the reason for the allocation. Wrap the statement in "quotes" if it contains any commas.	No
K	Allocation Template	If this block is being added to a device container with blockStatus=**Deployed**, the name of the allocation template to use to create address pools from the newly created block.	No
L	Interface Name	If this block is being added to a device container, the name of the interface to attach the block to.	Yes, if block is being added to device container. Otherwise, no.

Col	Field	Accepted Values	Required
M	Interface Offset or Address	The specific address(es), or offset(s) from the beginning, for the interface IP address(es). If an IP address is specified, it should be in the form xxx.xxx.xxx.xxx. If an integer is specified, it will be interpreted as an offset from the beginning of the block (i.e. an offset of 2 in a /24 block will create an interface xxx.xxx.xxx.2). This can also be the string "from-start" or "from-end" if you are attaching a block to a device container and wish IPAM to determine the first available address from the start or the end of the block. Multiple addresses can be specified by separating the values with the '|' character. For example, specify 3 interface offsets as 1|2|3. You can modify or delete interface addresses using the overwrite option. Specify the complete list of addresses using the IP address, separated by the '|' character if there is more than one. You cannot add an interface address or delete all interface addresses. Note that you cannot modify or delete a virtual interface address. Note that if you are changing the status of a block in a device container from **Reserved** to **Deployed**, an interface offset of 1 is assigned.	No. An offset of 1 is assumed if none is specified.
N	Create Reverse Domains	Whether or not to automatically create reverse DNS domain(s) for this block. Accepted values are **true** or **false**. If not specified, defaults to **false**.	No
O	Domain Type	Specify the domain type for the reverse DNS domain(s) to be created when Create Reverse Domains is "true". If not specified, defaults to "Default".	No
P	User Defined Fields	A series of name=value pairs, where the name is the UDF name and the value is desired value. Multiple pairs can be specified by separating each pair with the '|' character. For example, **UDFone=value one |UDFtwo=value two.** If the UDF type is Checkbox, the valid values are "on" or "off". If the UDF type is Textarea, use "\n" to separate lines. When this column contains the word "Expanded", an expanded format input file is expected. See above for an example of this format.	Yes, for UDFs defined as required fields.
Q	Exclude From Discovery	Flag indicating if this subnet should be included in Host Discovery tasks. Accepted values are **true** or **false**. If not specified, defaults to **false**. **Valid only for Deployed blocks.**	No
R	DHCP Option Set	The name of a DHCP Option Set defined within IPAM that should apply to this subnet. **Valid only for Deployed blocks.**	No
S	DHCP Policy Set	The name of a DHCP Policy Set defined within IPAM that should apply to this subnet. **Valid only for Deployed blocks.**	No

Col	Field	Accepted Values	Required
T	DNS Servers	The list of default DNS Servers for this subnet. This list is supplied to DHCP clients on this subnet. The server name or IP Address is valid. For multiple servers, separate the server names with a vertical bar (|). **Valid only for Deployed blocks.**	No
U	Default Gateway	The default gateway address for this subnet. This address is supplied to DHCP clients on this subnet. Specify a single IP address, or a comma-separated list surrounded by double-quotes. **Valid only for Deployed blocks.**	No
V	Primary DHCP Server	The name or IP Address of the primary DHCP server for this address space. **Valid only for Deployed blocks.**	No
W	Failover DHCP Server	The name or IP Address of the failover DHCP Server for this address space. **Valid only for Deployed blocks.**	No
X	DNS Forward Domains	The list of DNS Forward domains for this address space, separated by a vertical bar (|). This list will appear in the GUI when choosing domains for devices. To specify a domain type, specify the domain followed by '/' followed by the domain type. **Valid only for Deployed blocks.** For example: hr.ins.com.|dmz.com./External In this example, hr.ins.com uses the default domain type, and dmz.com is of type 'External'.	No
Y	DNS Reverse Domains	The list of DNS Reverse domains for this address space, separated by a vertical bar (|). This list will appear in the GUI when choosing domains for devices. To specify a domain type, specify the domain followed by '/' followed by the domain type. **Valid only for Deployed blocks.** For example: 0-15.1.0.10.in-addr.arpa. /External |40.10.in-addr.arpa. In this example, 0-15.1.0.10.in-addr.arpa. is of type 'External', and 40.0.10.in-addr.arpa. uses the default domain type.	No
Z	Primary WINS Server	The IP Address of the Primary WINS Server for this subnet. Used to provide this information to DHCP for Dynamic Address types. Multiple WINS servers may be specified, separated by a comma.	No
AA	Allocation Strategy	The Automatic Allocation Strategy to use where a block address is not provided. Valid options are: '*Bestfit*' (the **default** option when none is specified) '*Sparse*' (IPv6 only) '*Random*'. (IPv6 only) Note this field is not used for overwrite.	No
AB	Primary Subnet	Flag indicating if this subnet is primary. Accepted values are **true** or **false**. If not specified, defaults to **false**. **Valid only for Deployed blocks.**	No

Col	Field	Accepted Values	Required
AC	Network Link	**Valid for deployed blocks in logical containers only**, the name of a logical network link already defined in IPAM. This is the name of the Shared Network Segment for this subnet. This value is automatically assigned for blocks in device containers.	No
AD	NonBroadcast	Flag indicating if this is a Non-Broadcast subnet. Non-broadcast subnets are allowed to assign devices to the subnet and broadcast addresses. Accepted values are **true** or **false**. If not specified, defaults to **false.** **Valid for IPv4 Deployed blocks only.**	No
AE	CascadePrimary DHCPServer	If address pool or individual IP address objects within the subnet reference a specific DHCP server, this attribute can be used to allow the 'primaryDHCPServer' attribute value to "cascade" to these address pool and IP address objects. Specify **true** to apply this update. Note that address pool and IP address objects that are configured with "Same as Subnet" for the primary DHCP server will be unaffected. Applies to Overwrite only.	No

ImportContainer

Overview

The **ImportContainer** CLI allows the user to bulk import containers into IPAM.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.ImportContainerCLI -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportContainer.sh -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportContainer.cmd -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the file to import. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-imported) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example imports containers from the *newcontainers.csv* file, places into the *newcontainers.reject* file any records that could not be imported, and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/ImportContainer.sh -u joe -p joepwd -f newcontainers.csv
-r newcontainers.reject -e importerrors.txt
```

File Format

Col	Field	Accepted Values	Required
A	Container Name	The name of the container. If you are creating a device container, this container name must match exactly the name of a network element already in the database or the record will be rejected.	Yes
B	Container Description	A brief description of the container. Use "\n" to separate lines.	No

Col	Field	Accepted Values	Required
C	Parent Container	The name of the parent container for this container. Names can be in either short or long format. Short format example: **Dallas**. Long format example: **InControl/Texas/Dallas**. Long format eliminates ambiguity in cases where there are duplicate container names. If using the long format, the name must be the complete path beginning at the top of the container tree.	Yes
D	Container Type	Either logical or device.	Yes
E	Rule1	A listing of the valid block types for this container, separated by '/'. To specify information templates to be used for a block type, specify the block type followed by '|' followed by the information template name. For example: blocktype1|templateone/blocktype2/block type3|templatetwo In this example, blocktype2 does not use an information template.	No
F	Rule2	A listing of the block types enabled for root block creation, separated by '/'. Note this applies only to logical containers, and will be ignored if specified for device containers.	No
G	Rule3	A listing of the block types that can be used for space allocation from the parent container, separated by '/'.	No
H	Rule4	A listing of the block types for which SWIP Names are required, separated by '/'.	No
I	Rule5	A listing of the device types for this container, separated by '/'. To specify information templates to be used for a device type, specify the device type followed by '|' followed by the information template name. For example: devicetype1|templateone/devicetype2/ devicetype3|templatetwo In this example, devicetype2 does not use an information template. To specify that all device types should be allowed, use **ALL**. To specify that no device types should be allowed, use **NONE**. **ALL** is the default.	No
J	Information Template	A listing of pre-existing information templates to be associated with this container, separated by '|'.	No

Col	Field	Accepted Values	Required
K	User Defined Fields	Specify the values for the UDFs in the container information templates, specified in the previous parameter. Specify as a series of *name*=*value* pairs, where the *name* is the UDF name and the *value* is the desired value. Multiple fields can be specified by separating each name=value pair with the '|' character. For example: fieldOne=valueOne|fieldTwo=valueTwo If the UDF type is Checkbox, the valid values are "on" or "off". If the UDF type is Textarea, use "\n" to separate lines.	Yes, for UDFs defined as required fields
L	Maintain History Records	Specify whether or not Container History and Block History records will be kept for all appropriate block types. The history records are created each time the Global Utilization Rollup task is run. Accepted values are **true** or **false**. If not specified, defaults to **false**.	No

ImportDevice

Overview

The **ImportDevice** CLI imports devices into IPAM. This is used to bulk load information about existing network devices. It also allows modification of existing records when using the overwrite option (-o) and the expanded format, described later in this section.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.ImportDevice -u <userId> -p <pswd>
-f <import_file> [-e <error messages>] [-r <rejects file>] [-o] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportDevice.sh -u <userId> -p <pswd>
-f <import_file> [-e <error messages>] [-r <rejects file>] [-o] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportDevice.cmd -u <userId> -p <pswd>
-f <import_file> [-e <error messages>] [-r <rejects file>] [-o] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the file to import. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-imported) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.
-o	No	Overwrite option. Specify whether the import file contents will overwrite any matching records that exist in the database. This option requires expanded format. See below.
-v	No	Produces verbose output.

Usage Example

This example imports all of the devices in the file *devices.csv* and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/ImportDevice.sh -u joe -p joepwd -f devices.csv -e importerrors.txt
-r importrejects.txt
```

Import with overwrite using expanded format and the !BLANK! keyword

This example imports devices using the expanded format.

```
$INCHOME/etc/cli/ImportDevice.sh -u joe -p joepwd -f cexportdevice.csv -o
```

Here is the input file:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1	^ipaddres	addressTy	hostname	deviceTyp	hwType	MACAddr	resourceR	domainNa	container	domainTy	descriptio	userDefin	aliases	dupWarni	interfaces	excludeFr	virtual	DUID	relayAgen	relayAgen	firstName	lastName	title	location
2	10.0.0.1	Dynamic	PC00000	PC			TRUE	test.com.	East	Default		Expanded		FALSE	Default	FALSE	FALSE		1.12E+09	AABBCCDI	Charles	Black	Engineer	
3	10.0.0.2	Dynamic	PC00001	PC			TRUE	test.com.	East	Default		Expanded		FALSE	Default	FALSE	FALSE				Jane	Green	!BLANK!	
4	10.0.0.3	Static	Router00	Router	Ethernet	9.88E+08	TRUE	test.com.	East	Default				FALSE	Default	FALSE	FALSE							
5	10.0.0.4	Static	Router00	Router	Ethernet	98767645	TRUE	test.com.	East	Default		Expanded		FALSE	Default	FALSE	FALSE							Exton
6																								

Note the following:

- You can produce a file in this format using the ExportDevice CLI.
- The first row is a header line, beginning with "^". The column headers beyond the architected column T contain the user defined field tags. This row is required in order to modify devices via this CLI, regardless of whether or not the expanded format is being used.
- The –o (overwite) parameter is required in order to modify devices via this CLI.
- The rows describe devices to be either imported or modified. In other words, existing devices will be modified; new devices will be imported. Devices to be modified are identified by hostname, ipaddress or MACAddress.
- Column L contains "Expanded" when the row includes user defined fields beyond column T. For example, row 4 has no user defined fields defined for that device.
- Columns U-X contain the values for the user defined fields. If there is no value for a field in the device, an empty cell will be exported. For example, row 5 has no value in columns U-W.
- To clear a field in an existing record, specify "!BLANK!", as shown in row 3 column W. A cell with no value, as in row 4 column X, will result in no change to the value for the record stored in the database.
- If an expanded user defined field column contains data for a device where that field is not defined, an error will be reported.
- If your site supports duplicate hostnames, in order to import a new device with a hostname already in use, do not use the overwrite feature. Use import without overwrite, and specify ignore warnings (column N) as **true**.

File Format

Col	Field	Accepted Values	Required
A	IP Address	The IP Addresses of the Device. If there is more than one, the device is created as a multi-homed device. Separate multiple IP addresses with a vertical bar ("|"). For single-homed devices only, to indicate that IPAM should use the next available address in a block, specify the block address, followed by "/from-start" or "/from-end", for example: 10.30.0.0/from-start 3FFE::/from-start For devices with multiple IP Addresses on a single interface (devices with address type **Interface**), use one or more address to locate the device for an overwrite. Note that device attributes can be modified, but an Interface address cannot be modified or deleted. Use ImportChildBlock with overwrite to modify or delete an Interface address.	Yes
B	Address Type	The address type of this device. Accepted values are: **Static**, **Dynamic DHCP**, **Automatic DHCP**, **Manual DHCP** and **Reserved**. For DHCP V6: **Dynamic NA DHCPv6**, **Automatic NA DHCPv6**, **Manual NA DHCPv6**, **Dynamic TA DHCPv6** and **Automatic TA DHCPv6**. Note that if **Dynamic**, **Automatic** or **Manual DHCP** is specified, there must be a DHCP server defined in the subnet policies for this IP Address. Devices of type **Interface** may not be imported, but they can be modified using overwrite. However, the address type and IP address cannot be updated.	Yes
C	Host Name	Valid host name or **APPLYNAMINGPOLICY**	Yes
D	Device Type	The name of a device type configured in IPAM.	Yes, if Hostname specifies use of naming policy
E	Hardware Type	Specify **Ethernet** or **Token Ring**. When Hardware Type is specified, at least one MAC Address must also be specified. If MAC Address is specified, this will default to **Ethernet**.	No
F	MAC Address	The hardware MAC addresses of the device. Separate multiple entries with a vertical bar ("|"). If not left blank, there must be one MAC or a place-holder vertical bar for each IP Address in column A.	Yes, if Hardware Type is specified or if address type is Manual DHCP

Col	Field	Accepted Values	Required
G	Resource Record Flag	Whether or not to add resource records for this device. Accepted values are **true** or **false**. If not specified, defaults to **false**. Note that the domain name must be specified if the block policy has no forward domains. Also, the reverse domain must exist in order for the PTR record to be added.	No
H	Domain Name	Domain name already defined to IPAM.	Yes, if Resource Record Flag is "true" and the block policy has no forward domains.
I	Container	The name of the container that contains the block. Names can be in either short or long format. Short format example: **Dallas**. Long format example: **InControl/Texas/Dallas**. Long format eliminates ambiguity in cases where there are duplicate container names. If there is more than one IP Address, and the addresses are in different containers where overlapping space is used, you can specify multiple containers to resolve ambiguity. Separate multiple containers with a vertical bar ("|").	Yes, if overlapping space is in use and the block name is ambiguous.
J	Domain Type	Domain type name already defined to IPAM. If not specified, the "Default" domain type will be used	No
K	Description	A description of the device. Use "\n" to separate lines.	No
L	User Defined Fields	A series of *name*=*value* pairs, where the *name* is the UDF name and the *value* is desired value. Multiple fields can be specified by separating each name=value pair with the '|' character. For example, fieldOne=valueOne|fieldTwo=valueTwo. If the UDF type is Checkbox, the valid values are "on" or "off". If the UDF type is Textarea, use "\n" to separate lines. When this column contains the word "Expanded", an expanded format input file is expected. See above for an example of this format.	Yes, for UDFs defined as required fields.
M	Aliases	The alias or list of aliases for this hostname. When you specify an alias, a CNAME record is created. The alias may be fully qualified (contains a trailing dot), or not. When fully qualified, everything after the first qualifier is interpreted as a domain name. When not fully qualified, the CNAME record will be created in the same domain as the device. Specify multiple aliases by separating each one with the '|' character. To use this field, you must also specify Resource Record Flag = **true**.	No

Col	Field	Accepted Values	Required
N	Ignore Warning	If the administrator policy of the user indicates "Warn" for the "Allow Duplicate Hostnames Checking" option, the warning will be ignored and the device added with the duplicate hostname when this field is **true**. Also, if the administrator policy of the user indicates "Warn" for the "Allow Duplicate Hardware Address (MAC) Checking " option, the warning will be ignored and the device added with the duplicate hardware address when this field is **true**. Accepted values are **true** or **false**. If not specified, defaults to **false**.	No
O	Interface Names	Specify the names of the interfaces created for a multi-homed device. Separate entries with a vertical bar ("|"). There must one entry for each IP Address in Column A.	Yes, if multiple IP Addresses are entered in Column A.
P	Exclude from Discovery Flags	Flag indicating if this subnet should be included in Host Discovery tasks. Accepted values are **true** or **false**. If not specified, defaults to **false**. Specify the flags for each interface created for a multi-homed device. Separate entries with a vertical bar ("|"). There must one entry for each IP Address in Column A.	Yes, if multiple IP Addresses are entered in Column A.
Q	Virtual flag	Reserved and will be ignored	No
R	DUID	DHCP Unique Identifier	Yes, for Address Type "Manual NA DHCPV6".
S	Relay Agent Circuit Id	Populated after a DHCP Collections task is run.	Ignored
T	Relay Agent Remote Id	Populated after a DHCP Collections task is run.	Ignored

ImportDeviceResourceRecord

Overview

The **ImportDeviceResourceRecord** CLI allows the user to bulk import DNS resource records for a device into IPAM.

Usage

Direct

```
$INCHOME/jre/bin/java –cp $CLASSPATH
com.diamondip.ipcontrol.cli.ImportDeviceResourceRecordCLI –u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportDeviceResourceRecord.sh –u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportDeviceResourceRecord.cmd –u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the file to import. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-imported) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example imports resource records from the *newresourcerecs.csv* file, places into the *newresourcerecs.reject* file any records that could not be imported, and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/ImportDeviceResourceRecord.sh –u joe –p joepwd
–f newresourcerecs.csv –r newresroucerecs.reject –e importerrors.txt
```

File Format

Col	Field	Accepted Values	Required
A	Domain	The name of the domain to which the resource records will be added.	Yes
B	Domain Type	The name of the domain type to which the domain belongs. Defaults to "Default"	No
C	Owner	The OWNER section of the resource record. Note that this section is specific to the type of resource record. Refer to the appropriate RFC for exact text that should be entered.	Yes
D	Host Name	The device host name.	Yes, unless IP Address is specified.
E	IP Address	The IP Address of the Device.	Yes, unless Host Name is specified.
F	Container	The name of the container that holds the device. This is required only if there is overlapping address space in use and the IP address is in overlapping space. The container is then used to uniquely determine the device.	Yes, if IP Address in overlapping space.
G	TTL	The Time To Live.	No
H	Class	The value currently supported is **IN**. If not specified, defaults to **IN**.	No
I	Resource Record Type	The type of resource record being imported.	Yes
J	Data	The text for the DATA area of the resource record. Note that this section is specific to the type of resource record. Refer to the appropriate RFC for exact text that should be entered.	Yes
K	Comment	Text to be appended to the resource record.	No

ImportDhcpServer

Overview

The **ImportDhcpServer** CLI creates DHCP Servers in IPAM.

Usage

Direct

```
$INCHOME/jre/bin/java –cp $CLASSPATH
com.diamondip.ipcontrol.cli.ImportDhcpServer –u <userId> -p <pswd>
-f <import filename> [-e <error messages>] [-r <rejects file>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportDhcpServer.sh –u <userId> -p <pswd>
-f <import filename> [-e <error messages>] [-r <rejects file>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportDhcpServer.cmd –u <userId> -p <pswd>
–f <import filename> [-e <error messages>] [-r <rejects file>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the file to import. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-imported) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

File Format

Col	Field	Accepted Values	Required
A	Name	The name of the DHCP Server. This is often the hostname of the system running the server.	Yes
B	IP Address	The IP Address of the DHCP Server. This must be a legal IP address. If the V4V6Both parameter is V6, then this must be a legal IPv6 address.	Yes
C	Product	The product name of the DHCP Server. This must be one of the products defined in IPAM.	Yes
D	Agent	The IPAM agent that manages the server	Yes
E	Default Threshold	The default alert threshold applied to all scopes managed by this server. This must be a number between 0 and 100. Defaults to 90.	No
F	Global Sync	Specify TRUE to include this server in Global Sync tasks. Defaults to false.	No
G	Configuration Path	The path to the server's configuration file. Must be a legal, fully qualified path name for the host system.	No

Col	Field	Accepted Values	Required
H	Lease Path	The path to the lease file. Must be a legal, fully qualified path for the host system. Must be a legal, fully qualified path name for the host system.	No
I	Start Script	The path to the script that starts the server. Must be a legal, fully qualified path name for the host system.	No
J	Stop Script	The path to the script that stops the server. Must be a legal, fully qualified path name for the host system.	No
K	Collection Type	SCP or FTP	No
L	Collection Port	Must be between 1 and 65535. Defaults to 21 for FTP and 22 for SCP.	No
M	Collection User	User name for SCP/FTP access to the executive.	No
N	Collection Password	Password for the collection user.	No
O	Collect Backup Subnets	Specify TRUE to collection statistics on backup subnets. Defaults to FALSE.	No
P	CLI Command	Collection program name	No
Q	CLI User	Collection program user credential.	No
R	CLI Password	Collection program password credential	No
S	CLI Arguments	Arguments to the Collection program. Differs according to product.	No
T	DDNS	Specify 'interim', 'standard', or 'lastin' to enable dynamic DNS updates when this server issues a lease. Defaults to 'none'.	No
U	DHCP Option Set	The name of an option set defined in IPAM.	No
V	DHCP Policy Set	The name of a policy set defined in IPAM.	No
W	DHCP Client Classes	The names of client classes defined in IPAM that this server will be using. Separate multiple client classes with a vertical bar ("|").	No
X	DHCP Failover IP Address	The IP Address used by the DHCP server for failover communications.	No
Y	DHCP Failover Port	The Port used by the DHCP server for failover communications.	No
Z	Configuration File Pre-Extension	Text to prepend to the DHCP server configuration file. This can be the text itself, or a reference to a file. If the field begins with "file:", then the remainder of the field is treated as a file name and the file's contents are used.	No
AA	Configuration File Post-Extension	Text to append to the DHCP server configuration file. This can be the text itself, or a reference to a file. If the field begins with "file:", then the remainder of the field is treated as a file name and the file's contents are used.	
AB	V4V6Both	IP version supported by the DHCP server. Valid values are 'V4', 'V6' or 'Both'	Yes

ImportDNS

Overview

The **ImportDNS** CLI allows the user to import the contents of a DNS zone file, or the zone files referenced by master zones declared in an ISC BIND 8.x and newer *named.conf* file.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.dnsimport.DNSImport -f <import filename> -s <server>
[-v <view>] [-z <zone>] [-l] [-t <domainType>] [-m <view/zone=domainType, … >]
[-n] [-2 None|ZoneOnly|ZoneAndRR] [-c container]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportDNS.sh -f <import filename> -s <server> [-v <view>] [-z <zone>] [-l]
[-t <domainType>] [-m <view/zone=domainType, … >] [-n] [-2 None|ZoneOnly|ZoneAndRR]
[-c container]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportDNS.cmd -f <import filename> -s <server> [-v <view>] [-z <zone>]
[-l] [-t <domainType>] [-m <view/zone=domainType, … >] [-n] [-2 None|ZoneOnly|ZoneAndRR]
[-c container]
```

Parameters

Parameter	Required	Description
-f <import filename>	Yes	File name containing data to import. If the -z parameter is not supplied, the file is assumed to be a ISC BIND 8.x or newer configuration file. Otherwise it is assumed to be a ISC DNS zone file.
-s <server>	Yes	The name of the DNS network service as defined in IPAM to import the zone data into.
-v <view>	No	The name of the view in which new domains should be created. If supplied the view must exist. If not supplied new domains will be created in the view named 'Default.'
-z <zone>	No	The name of the zone. Must be supplied when importing a single zone file. See -f.
-l	No	Import "flat" zone. The domain hierarchy will be created to support the domain in the SOA, but any other sub-domains found within the zone will not be created as separate domains.
-t	No	The name of the DomainType to assign to the imported domain(s). If not specified, then the Default DomainType is used.

Parameter	Required	Description

-m	No	Comma separated list of ViewName/ZoneName=DomainType entries. These are used as an override of either the –t option or the system Default DomainType.
		Note: This option is only valid when parsing a configuration file, not when parsing an individual zone file using the –z option.
		An example with Views defined:
		`-m "internal/hr.ins.com=internal,external/dmz.ins.com=external"`
		An example when views are not defined:
		`-m "hr.ins.com=internal,dmz.ins.com=external"`
		Note: Quotes are not necessary unless there are spaces in the DomainType names. They are used here for completeness.

-n	No	Allow NS records to be imported. By default, NS records are ignored. By specifying this option, NS records will be imported into the domain Resources Records. In addition, the Auto-generate NS records flag for the zone will be turned off.
-2	No	Slave (or secondary) zone handling. Valid values are: None – (default) Ignore all slave zones. ZoneOnly – Import the zone definition only. ZoneAndRR – Import the zone definition and the Resource Records in the zone file. **Note:** This option is only valid when parsing a configuration file, not when parsing an individual zone file using the –z option.
-c	No	Container name. The *full pathname* (e.g. "InControl/North America/US/PA") of a container in IPAM to be used to distinguish between overlapping address blocks. When ImportDNS encounters A or PTR records when importing a zone, it will attempt to create Device and IP Address records accordingly. If the IP reflected in the 'rdata' of an A record or the 'owner' of a PTR record falls within an In-Use/Deployed Block which overlaps space with another In-Use/Deployed Block in another portion of the container hierarchy, then this parameter can be used to specify a container which is used to select the proper block .
-h, -?	No	Print help.

Description

The **`ImportDNS`** CLI imports DNS resource records contained in zone data files into IPAM. It does this by sequentially reading each resource record contained in a specified zone file, processing each one according to a set of rules described below, and then inserting some portion of the resulting data into IPAM.

Resource records read from zone files are initially processed using the same rules described in RFC 1035. This means that after initial processing the resource record will contain all the required fields: Name, Type, Class, TTL, and Rdata. This processing can include filling in any missing TTL values or Name fields, correctly using the Origin when an @ character is

encountered in a name field, and appending the Origin when appropriate as described by RFC 1035. The resulting resource record is then processed using rules specific to each resource record's Type field. The type-specific rules are described in Table 2:

Table 2 Type-specific Rules

Type	Description
SOA	Data from the SOA record is used to create or update a domain in IPAM. The domain name is taken from the name field of the resource record. If the domain does not exist in IPAM it will be created. When a domain is created by the **`ImportDNS`** CLI its parent domain will be created and linked to its child, or sub-domain. This process continues until an existing parent is found, or a top level domain is created. Top level domains created in this way have no parent associated with them. After the domain and its parents have been created, or if it already exists in IPAM, the data from the imported record will be used to update the domain. This includes the serial number, refresh, retry, expire, negative caching TTL, default TTL, and contact fields. In addition, the managed and delegated properties will be set for the domain. Note: If parent domains are created as a result of the record being imported, the delegated property will be set, while the managed property will not be set. If the domain name ends with *.in-addr.arpa.* the reverse property will be set.
A	Data from each A record is used to create resource record entries attached to domains in IPAM and additionally can create individual IP addresses and devices. When importing an A record the **`ImportDNS`** CLI will use the left-most label of the domain supplied in the name field as the host portion of the FQDN for the host. If the resulting domain (everything to the right of the left-most label) does not exist in IPAM, then it will be created along with any necessary parent domains. The newly created domain will have its managed property set, while its delegated property will not be set. If parent domains were also created, their delegated property will be set and their managed property will not be set. If the address found in the rdata field of the resource record can be located within an existing address block defined in IPAM, an IP address will be created using the rdata field, and a device will be created using the name field. The device and IP address will also be associated with the resource record in addition to the domain.
PTR	Data from each PTR record is used in the same way as data from A records with the exception that the name and rdata field are swapped when creating an IP address and device.
NS	NS records are ignored by the **`ImportDNS`** CLI.
MX	MX records are imported into IPAM and attached to the domain supplied in the name field.

Type	Description
SRV	The data from SRV records are processed in the same way as other resource records with the exception of the name field. The name field of SRV records specifies a service available for the domain for which it is a part of. The service type and protocol is encoded in the left portion of its name field. To avoid collision with the rest of the domain name space, leading underbars are prepended to the labels that describe the service. This practice is not always followed in the field and so the **ImportDNS** CLI uses the following rule to determine where the domain name part of the name field starts. It considers all the labels to the right of the right-most label that starts with an underscore to be part of the domain name of the SRV record. For example in the following SRV record: _ldap._tcp.pdc._msdcs.sw.ins.com. 600 SRV 0 100 400 pdc.sw.ins.com. The service specification part would be: _ldap._tcp.pdc._msdcs and the domain name part would be: sw.ins.com.
All others	The domain name that the resource record will be placed in is taken from the name field of the resource record after the left label has been removed. If the domain cannot be found in IPAM it will be created, along with any necessary parent domains. Parent domains will have the delegated property set and the managed property not set. A resource record object is created using the data supplied by the imported record, and it is linked to the domain.

Examples

The usage examples below use these example zone data or configuration files:

Example zone data file *db.example*:

```
$TTL 3600
@ IN SOA thomas.example.com. hostmaster.thomas.example.com. (
                              1       ; Serial number
                              10800   ; Refresh
                              3600        ; Retry
                              604800  ; Expire
                              86400 )     ; Minimum TTL
IN NS                       thomas.example.com.
localhost                   IN A 127.0.0.1
dhcp                            IN A 192.168.0.1
thomas                      IN A 192.168.0.2
msdc                            IN A 192.168.0.3
www                             IN A 192.168.0.4
ftp                             IN A 192.168.0.5
mail                            IN A 192.168.0.6

_ftp._tcp                   IN SRV   0 100 400 ftp.example.com.
_http._tcp                  IN SRV  0 100 434.www.example.com.

dns                         IN CNAME thomas.example.com
w3                          IN CNAME www.example.com.
web                         IN CNAME www.example.com.
incoming                    IN CNAME ftp.example.com.
smtp                        IN CNAME mail.example.com.
pop                         IN CNAME mail.example.com.
```

Example zone data file *db.192.168.0*:

```
0.168.192.in-addr.arpa IN SOA thomas.example.com. hostmaster.thomas.example.com.  (
1                           ; Serial number
                            10800   ; Refresh
                            3600        ; Retry
                            604800  ; Expire
                            86400 )     ; Minimum TTL

2.0.168.192.in-addr.arpa.  IN NS thomas.example.com.

1.0.168.192.in-addr.arpa.  PTR dhcp.example.com.
2.0.168.192.in-addr.arpa.  PTR thomas.example.com.
3.0.168.192.in-addr.arpa.  PTR msdc.example.com.
4.0.168.192.in-addr.arpa.  PTR www.example.com.
5.0.168.192.in-addr.arpa.  PTR ftp.example.com.
6.0.168.192.in-addr.arpa.  PTR mail.example.com.
```

Example Bind 9 configuration file *named.conf*:

```
Options {
                          directory "/var/lib/named";
                          notify no;
};

zone "example.com." {
                          type master;
                          file "db.example";
};

zone "0.168.192.in-addr.arpa." {
                          type master;
                          file "db.0.168.192";
};
```

Usage Example 1

This example creates master zones linked to the server `dns1.sw.ins.com` using the zone data contained within the zone files that are referenced by the master zone declarations within the *named.conf* file, specifically the files *db.example* and *db.0.168.192*. The following domains are created: `com.`, `example.com.`, and `0.168.192.in-addr.arpa.` The domain `example.com.` has 14 resource records associated with it, all the ones declared in the *db.example* file except the SOA record, and the NS record. The data from the SOA record updates the domain *example.com* with the values from its rdata field. The domain `example.com.` is marked as delegated and managed. The domain `0.168.192.in-addr.arpa.` has associated with it the resource records from the *db.0.168.192* file except the SOA and NS records. Its properties are also updated with the information from the SOA record for the zone, and are marked as delegated and managed. It is also marked as reverse.

```
$INCHOME/etc/cli/ImportDNS.cmd -f /etc/named.conf -s dns1.sw.ins.com
```

Usage Example 2

This example imports the resource records declared in the zone file *db.192.168.0*. The domain `0.168.192.in-addr.arpa` is created if it does not already exist. The domain has its delegated, managed, and reverse properties set. The zone `0.168.192.in-addr.arpa` is created and associated with the server `dns1.sw.ins.com`.

```
$INCHOME/etc/cli/ImportDNS.cmd -f db.192.168.0 -s dns1.sw.ins.com
-z 0.168.192.in-addr.arpa
```

Usage Example 3

This example imports the resource records declared in the zone file *db.hr.ins.com*. The domain `hr.ins.com` is created if it does not already exist. The domain has its delegated, managed, and reverse properties set. The domain is assigned to the "internal" DomainType. The zone `hr.ins.com` is created and associated with the server `dns1.sw.ins.com`.

```
$INCHOME/etc/cli/ImportDNS.cmd -f db.hr.ins.com -s dns1.sw.ins.com
-z hr.ins.com –t internal
```

Usage Example 4

This example creates master zones linked to the server `dns1.sw.ins.com` using the zone data contained within the zone files that are referenced by the master zone declarations within the *named.conf* file, specifically the files *db.example* and *db.0.168.192*. The following domains are created, if they did not already exist: `com.`, `example.com.`, `in-addr.arpa.`, and `0.168.192.in-addr.arpa.` The domain `example.com.` has 14 resource records associated with it, all the ones declared in the *db.example* file except the SOA record, and the NS record. The data from the SOA record update the domain `example.com` with the values from its rdata field. The domain `example.com.` is marked as delegated and managed. The domains `com.` and `example.com` are assigned to the "external" DomainType. The domains `in-addr.arpa` and `0.168.192.in-addr.arpa.` are assigned to the "Default" DomainType. The domain `0.168.192.in-addr.arpa.` has associated with it the resource records from the *db.0.168.192* file except the SOA and NS records. Its properties are also updated with the information from the SOA record for the zone, and are marked as delegated and managed. It is also marked as reverse.

```
$INCHOME/etc/cli/ImportDNS.cmd -f /etc/named.conf -s dns1.sw.ins.com
–m "example.com=external"
```

Usage Example 5

This example creates master zones linked to the server `dns1.sw.ins.com` using the zone data contained within the zone files that are referenced by the master zone declarations within the *named.conf* file, specifically the files *db.example* and *db.0.168.192*. The following domains are created: `com.`, `example.com.`, and `0.168.192.in-addr.arpa.` The domain `example.com.` has 14 resource records associated with it, all the ones declared in the *db.example* file except the SOA record, and the NS record. The data from the SOA record updates the domain `example.com` with the values from its rdata field. The domain `example.com.` is marked as delegated and managed. The domain `0.168.192.in-addr.arpa.` is associated with it the resource records from the *db.0.168.192* file except the SOA and NS records. Its properties are also updated with the information from the SOA record for the zone, and are marked as delegated and managed. It is also marked as reverse.

```
$INCHOME/etc/cli/ImportDNS.cmd -f /etc/named.conf -s dns1.sw.ins.com -2 ZoneOnly
```

In addition, a zone definition is created for the Slave zone `foo.com`. Its "masters" sub-statement is set to `192.168.0.1; 192.168.0.2`. If necessary the domain `foo.com` is

also created with the SOA information found in the file *bak.foo.com*. The domain is marked as delegated and managed. No resource records are imported for `foo.com`.

Usage Example 6

This example creates master zones linked to the server `dns1.sw.ins.com` using the zone data contained within the zone files that are referenced by the master zone declarations within the *named.conf* file, specifically the files *db.example* and *db.0.168.192*. The following domains are created: `com.`, `example.com.`, and `0.168.192.in-addr.arpa.` The domain `example.com.` has 14 resource records associated with it, all the ones declared in the *db.example* file except the SOA record, and the NS record. The data from the SOA record updates the domain `example.com` with the values from its rdata field. The domain `example.com.` is marked as delegated and managed. The domain `0.168.192.in-addr.arpa.` has associated with it the resource records from the *db.0.168.192* file except the SOA and NS records. Its properties are also updated with the information from the SOA record for the zone, and are marked as delegated and managed. It is also marked as reverse.

```
$INCHOME/etc/cli/ImportDNS.cmd -f /etc/named.conf -s dns1.sw.ins.com -2 ZoneAndRR
```

In addition, a zone definition is created for the Slave zone `foo.com`. Its "masters" sub-statement is set to `192.168.0.1; 192.168.0.2`. If it did not already exist, the domain `foo.com` is also created with the SOA information found in the file `bak.foo.com`. The domain is marked as delegated and managed. The domain has associated with it the resource records from the `bak.foo.com` file except the SOA and NS records.

Return codes

The **`ImportDNS`** CLI always returns 0.

ImportDomain

Overview

The **ImportDomain** CLI imports domains into IPAM.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.ImportDomain -u <userId> -p <pswd>
-f <import_file> [-e <error messages>] [-r <rejects file>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportDomain.sh -u <userId> -p <pswd>
-f <import_file> [-e <error messages>] [-r <rejects file>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportDomain.cmd -u <userId> -p <pswd>
-f <import_file> [-e <error messages>] [-r <rejects file>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the file to import. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-imported) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.
-v <verbose>	No	Produces verbose output.

Usage Example

This example imports all of the domains in the file *domains.csv* and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/ImportDomain.sh -u joe -p joepwd -f domains.csv -e importerrors.txt
-r importrejects.txt
```

File Format

Col	Field	Accepted Values	Required
A	Domain Name	The name of the domain being created.	Yes
B	Domain Type	Domain type name already defined to IPAM. If not specified, the "Default" domain type will be used. When importing an "ALIAS" domain its domain type is used to help locate the named (Column G) "TEMPLATE" domain it will be aliased to. Therefore they MUST match for the "TEMPLATE" domain to be found.	No

Col	Field	Accepted Values	Required
C	Managed	Indicates that this domain is fully defined in IPAM. Accepted values are **true** or **false**. If not specified, defaults to **true**.	No
D	Delegated	Indicates that this domain will be associated directly with a zone file. Accepted values are **true** or **false**. If not specified, defaults to **true**.	No
E	Reverse	Indicates that this domain is a reverse 'in-addr.arpa' or 'ip6.arpa' domain. Accepted values are **true** or **false**. If not specified, defaults to **false**.	No
F	Derivative	Specify the role of this domain. This can be one of "STANDARD", "TEMPLATE", or "ALIAS". If not specified, defaults to "STANDARD". The *delegated* flag MUST be set to true in order to import a ""TEMPLATE" or "ALIAS" domain.	No
G	Template Domain	Name of template domain. When importing an "ALIAS" domain its domain type (Column B) is used to help locate the "TEMPLATE" domain it will be aliased to. Therefore they MUST match for the template domain to be found.	Yes, if Derivative is "ALIAS"
H	Serial Number	Zone serial number. If not specified, defaults to "1"; ignored if Managed is "False".	No
I	Refresh	Zone refresh interval. If not specified, defaults to "10800"; ignored if Managed is "False".	No
J	Retry	Zone retry interval. If not specified, defaults to "3600"; ignored if Managed is "False".	No
K	Expire	Zone expire time. If not specified, defaults to "604800"; ignored if Managed is "False".	No
L	Negative Cache TTL	Zone negative cache time to live. If not specified, defaults to "86400"; ignored if Managed is "False".	No
M	Default TTL	Default time to live. If not specified, defaults to "86400"; ignored if Managed is "False".	No
N	Contact	The contact email address in dotted format. For example, an email address of 'root@ins.com', would be represented as 'root.ins.com'. If not specified a default contact name will be formed by prepending 'dnsadmin' to the domain name as in 'dnsadmin.ins.com.'.	No
O	Information Template	Pre-defined information template to be associated with this domain.	No
P	User Defined Fields	A series of *name*=*value* pairs, where the *name* is the UDF field name/tag and the *value* is desired value. Multiple fields can be specified by separating each name=value pair with the '|' character. For example, fieldOne=valueOne|fieldTwo=valueTwo. If the UDF type is Checkbox, the valid values are "on" or "off". If the UDF type is Textarea, use "\n" to separate lines.	Yes, for UDFs defined as required fields.

ImportDomainResourceRecord

Overview

The **ImportDomainResourceRecord** CLI allows the user to bulk import DNS resource records for a domain into IPAM. This CLI allows the administrator to enter resource records that are not bound to a particular device.

Note: For Glue records that link one zone to another, use the **ImportZoneResourceRecord** CLI.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.ImportDomainResourceRecordCLI -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportDomainResourceRecord.sh -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportDomainResourceRecord.cmd -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the file to import. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-imported) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example imports resource records from the n*ewresourcerecs.csv* file, places into the *newresourcerecs.reject* file any records that could not be imported, and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/ImportDomainResourceRecord.sh -u joe -p joepwd -f newresourcerecs.csv
-r newresroucerecs.reject -e importerrors.txt
```

File Format

Col	Field	Accepted Values	Required
A	Domain	The name of the domain to which the resource records will be added.	Yes
B	Domain Type	The name of the domain type to which the domain belongs. Defaults to "Default"	No
C	Owner	The OWNER section of the resource record. Note that this section is specific to the type of resource record. Refer to the appropriate RFC for exact text that should be entered.	Yes
D	TTL	The Time To Live.	No
E	Class	The value currently supported is **IN**. If not specified, defaults to **IN**.	No
F	Resource Record Type	The type of resource record being imported.	Yes
G	Data	The text for the DATA area of the resource record. Note that this section is specific to the type of resource record. Refer to the appropriate RFC for exact text that should be entered.	Yes
H	Comment	Text to be appended to the resource record.	No
I	Device-bound Record	When **true**, this indicates that the resource record is bound to a device. When **false**, this indicates that the resource record is associated with the domain only, and not a specific device. Valid on export only.	No

ImportElementSnapshot

Overview

The **`ImportElementSnapshot`** CLI allows the user to import interfaces and blocks discovered by either a "Discover Router Subnets" or "Global Synchronization of Network Elements" task. Typically, such tasks are used to query the current state of the network and perform difference analysis to compare actual deployment with the topology modeled in IPAM. However, during the initial setup of the system the queried information from these tasks can be used to populate the original IPAM model. This CLI then, is used to perform this initial population of discovered blocks and interfaces.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.ImportElementSnapshot -u <userId> -p <pswd>
-t <taskId> [-o <output file>] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportElementSnapshot.sh -u <userId> -p <pswd>
-t <taskId> [-o <output file>] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportElementSnapshot.cmd -u <userId> -p <pswd>
-t <taskId> [-o <output file>] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-t <taskId>	Yes	The Id of the "Discover Router Subnets" or "Global Synchronization of Network Elements" task.
-o <output file>	No	The name of the file to write all output. If not specified output will be sent to STDOUT.
-v	No	Produces verbose output.

Usage Example

This example imports blocks and interfaces discovered by task 12345, and places into the *elementsnapshot.out* file any output messages.

```
$INCHOME/etc/cli/ImportElementSnapshot.sh -u joe -p joepwd -t 12345 -o elementsnapshot.out
```

ImportGalaxyDomain

Overview

The **ImportGalaxyDomain** CLI allows the user to bulk import Galaxy Domains into IPAM.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.ImportGalaxyDomainCLI -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportGalaxyDomain.sh -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportGalaxyDomain.cmd -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the file to import. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-imported) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.
-v	No	Produces verbose output.

Usage Example

This example import galaxy domains from the *newgdomains.csv* file, places into the *newgdomains.reject* file any records that could not be imported, and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/ImportGalaxyDomain.sh -u joe -p joepwd -f newgdomains.csv
-r newgdomains.reject -e importerrors.txt
```

File Format

Col	Field	Accepted Values	Required
A	Galaxy Name	Name of the Galaxy to which to assign this domain.	Yes
B	Domain Name	Domain name, already defined to IPAM, to be assigned to specified galaxy.	Yes
C	Domain Type	The name of the domain type to which the domain belongs. If not specified, the "Default" domain type will be used.	No
D	View	The name of the galaxy view to which to assign this domain. If specified, the view must exist. If not specified, the new zone will be created in the view named 'GalaxyDefault.'	No

ImportNetElement

Overview

The **`ImportNetElement`** CLI allows the user to bulk import network elements into IPAM.

Note: This CLI has been deprecated as of IPAM 8.1.3. Use the **ImportNetworkElement** CLI instead. ImportNetElement support will be removed in a future release.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.ImportNetElementCLI -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportNetElement.sh -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportNetElement.cmd -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the file to import. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-imported) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example imports Network Elements from the *netelements.csv* file, places into the *netelements.reject* file any records that could not be imported, and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/ImportNetElement.sh -u joe -p joepwd -f netelements.csv
-r netelements.reject -e importerrors.txt
```

File Format

Col	Field	Accepted Values	Required
A	Name	The name of the Network Element. This can be any combination of letters and numbers.	Yes
B	IP Address/FQDN	The IP address or fully-qualified domain name (FQDN) of the Network Element. This must be a valid IPv4 or IPv6 IP address, or a fully-qualified host name.	Yes

Col	Field	Accepted Values	Required
C	Vendor	The vendor of the Network Element. Vendor must be predefined in IPAM. If not specified, defaults to **Unknown**.	Yes when Model is specified.
D	Model	The model name of the Network Element. Model must be predefined in IPAM. If not specified, defaults to **Unknown**.	Yes when Vendor is specified.
E	Type	The type of Network Element. Accepted values are **cmts**, **router**, **switch**, or **vpn**.	Yes
F	Global Sync	Whether or not to include this Network Element in the Global Sync process. Accepted values are **True** or **False** (case insensitive).	Yes
G	Agent Name	The exact name of the IPAM Agent that's responsible for contacting this Network Service.	Yes
H	Telnet User	A user name used to telnet to this device.	No
I	Telnet Password	A password used by the telnet user to telnet to this device.	No
J	Enable Password	Password used to enter "enabled" or "privileged" mode on the device.	No
K	Read Community String	The community string used by SNMP to read details from this network element. Set this when using SNMP V1. Otherwise use V3 parameters below.	No
L	Interface List	Separated by vertical bars ("|").	No
M	V3 Username	Required if using SNMP V3.	No
N	V3 Authentication Protocol	Either **MD5** or **SHA1**. Leave blank or set to **NONE** if no authentication.	No
O	V3 Authentication Password	Required if field N is set to either **MD5** or **SHA1**	No
P	V3 Privacy Protocol	Only **DES** supported at this time. Leave blank or set to **NONE** if no privacy.	No
Q	V3 Privacy Password	Required if field P is set to **DES**.	No
R	V3 Context Name	SNMP V3 Context name, if needed.	No
S	V3 Engine ID	SNMP V3 Engine ID, if needed.	No

ImportNetworkElement

Overview

The **ImportNetworkElement** CLI allows the user to bulk import network elements into IPAM. It deprecates the ImportNetElement CLI by by adding support for new fields and capabilities, most notably multiple agents and the ability to apply device interface templates on import. Telnet Username, Telnet Password, and Enable Password are not implemented as these fields are now deprecated.

This CLI also supports updating network elements via the overwrite parameter (-o).

Usage

Direct

```
$INCHOME/jre/bin/java –cp $CLASSPATH
com.diamondip.netcontrol.cli.ImportNetworkElementCLI –u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportNetworkElement.sh –u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportNetworkElement.cmd –u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the file to import. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-imported) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.
-o	No	Overwrite option. Specify whether the import file contents will overwrite any matching records that exist in the database. See below.
-v	No	Produces verbose output, including the name of the network element added or updated.

Usage Example

This example imports Network Elements from the *networkelements.csv* file, places into the *networkelements.reject* file any records that could not be imported, and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/ImportNetworkElement.sh –u joe –p joepwd –f networkelements.csv
–r networkelements.reject –e importerrors.txt
```

Import with overwrite

This example imports network elements with the overwrite option:

```
$INCHOME/etc/cli/ImportNetworkElement.sh –u joe –p joepwd –f networkelements.csv –o
```

Note the following:

- You can produce a file in the required format using the ExportNetworkElement CLI.
- The –o (overwrite) parameter is required in order to modify network elements via this CLI.
- The input file rows describe network elements to be either imported or modified. In other words, existing network elements will be modified; new network elements will be imported. Network elements to be modified are identified by network element name.
- An empty cell indicates no change will be made for that field.
- To clear a non-required field in an existing record, specify "!BLANK!".

File Format

Col	Field	Accepted Values	Required
A	Name	The name of the Network Element. This can be any combination of letters and numbers.	Yes
B	Description	A description of the Network Element. Use "\n" to separate lines.	No
C	IP Address	The IP address of the Network Element. This must be a valid IPv4 or IPv6 address.	No
D	Type	The device type of Network Element. Typical values include **CMTS**, **Router**, **Switch** and **VPN**, but any valid device type configured in IPAM is accepted.	No; defaults to **Unknown** .
E	Vendor	The vendor of the Network Element. Vendor must be predefined in IPAM. If not specified, defaults to **Unknown**.	Yes when **Model** is specified.
F	Model	The model name of the Network Element. Model must be predefined in IPAM. If not specified, defaults to **Unknown**.	Yes when **Vendor** is specified.
G	Agent Name(s)	The name(s) of the IPAM Agent(s) that are responsible for contacting this Network Element. If multiple agents are specified, separate their names with vertical bars("|").	No
H	Global Sync	Specifies whether or not to include this Network Element in the Global Sync process. Accepted values are **true** or **false**.	No; defaults to **false**
I	SNMP Version	Specify **V1, V2** or **V3**.	No; defaults to **V1**.
J	Read Community String	The community string used by SNMP to read details from this network element. Set this when using SNMP V1 or SNMP V2. Otherwise use the V3 parameters below.	Yes when **SNMP Version V2** is chosen.

Col	Field	Accepted Values	Required
K	SNMP Timeout	Specifies the number of milliseconds the SNMP Agent will wait without receiving any messages from its partner before it assumes that the connection to its partner has failed.	No
L	SNMP Retries	Specifies the max number of connection retries that the SNMP Agent will attempt.	No
M	V3 Username	User name for SNMP V3.	Yes for V3
N	V3 Authentication Protocol	Specify **MD5** or **SHA1**. Leave blank to disable authentication.	No
O	V3 Authentication Password	Required if authentication enabled. (Required on import only; optional on overwrite.)	Yes when authentica-tion enbled
P	V3 Privacy Protocol	Specify **AES** or **DES**. Leave blank to disable privacy protection. Also requires authentication to be active to be recognized.	No
Q	V3 Privacy Password	Required if privacy protection enabled. (Required on import only; optional on overwrite.)	Yes when privacy enbled
R	V3 Context Name	SNMP V3 context name, if needed.	No
S	V3 Engine ID	SNMP V3 engine ID, if needed.	No
T	Interface List	Specify interface name/status pair(s) separated by vertical bars ("|"). Specify status as **enabled**, **disabled** or **deploying**. For example: Ethernet0/enabled | Ethernet1/disabled If the status is not specified, it will default to **enabled**.	No; but if specified this data mutually exclusive relative to Column U.
U	Device Interface Template and Interface Status	Specify the name of the template to apply, and the status for the interfaces, separated by a '/'. Specify status as **enabled**, **disabled** or **deploying**. For example: TemplateOne/disabled. If the status is not specified, it will default to **enabled**.	No; but if specified this data mutually exclusive relative to Column T.
V	SNMP sysName	Populated after a Discover Router Subnets task is run against the network element. This is a standard MIB-II variables (see RFC-1213).	Ignored
W	SNMP sysDescr	Populated after a Discover Router Subnets task is run against the network element. This is a standard MIB-II variables (see RFC-1213).	Ignored
X	SNMP sysLocation	Populated after a Discover Router Subnets task is run against the network element. This is a standard MIB-II variables (see RFC-1213).	Ignored
Y	SNMP sysServices	Populated after a Discover Router Subnets task is run against the network element. This is a standard MIB-II variables (see RFC-1213).	Ignored

ImportNetElementInterface

Overview

The **ImportNetElementInterface** CLI allows the user to bulk import network element interfaces into IPAM.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.ImportNetElementInterfaceCLI -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportNetElementInterface.sh -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportNetElementInterface.cmd -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the file to import. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-imported) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.
-v	No	Produces verbose output.

Usage Example

This example imports Network Elements interfaces from the *netelementinterfaces.csv* file, places into the *netelementinterfaces.reject* file any records that could not be imported, and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/ImportNetElementInterface.sh -u joe -p joepwd
-f netelementinterfaces.csv -r netelementinterfaces.reject -e importerrors.txt
```

File Format

Col	Field	Accepted Values	Required
A	Name	The name of a Network Element already defined to IPAM.	Yes
B	Interface Name	The name of the interface being imported.	Yes
C	Status	The status of the new interface. This can be one of "Disabled", "Enabled", or "Deployed". The default is "Enabled".	No

ImportNetService

Overview

The **ImportNetService** CLI allows the user to bulk import DHCP network services into IPAM.

Note: This CLI has been deprecated as of IPAM 8.1.2. Use the **ImportDhcpServer** CLI instead. ImportNetService support will be removed in a future release.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.ImportNetServiceCLI -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportNetService.sh -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportNetService.cmd -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameter

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the file to import. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-imported) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example imports network services from the *netservices.csv* file, places into the *netservices.reject* file any records that could not be imported, and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/ImportNetService.sh -u joe -p joepwd -f netservices.csv
-r netservices.reject -e importerrors.txt
```

File Format

Col	Field	Accepted Values	Required
A	Name	The name of the Network Service. This can be any combination of letters and numbers.	Yes
B	IP Address/FQDN	The IP address or fully-qualified domain name (FQDN) of the Network Service. This must be a valid IPv4 or IPv6 IP address, or a fully-qualified host name.	Yes

Col	Field	Accepted Values	Required
C	Type	The type of Network Service. Accepted value is **dhcp**. If this column is left blank, **dhcp** is assumed.	No
D	Product name	The Network Service product name. This must be a value already defined in IPAM, for example, **CNR DHCP**.	Yes
E	Agent name	The name of the IPAM Agent that's responsible for contacting this Network Service.	Yes
F	Global Sync	Whether or not to include this Network Service in the Global Sync process. Accepted values are **True** or **False** (case insensitive). If no value is specified, this will default to **False**.	No
G	Collection Method	The method by which the IPAM Agent will collect data from the Network Service. Accepted values are **scp** or **ftp**. For CNR servers, the collection type may also be **cnrsdk** for those servers that are managed via the SDK.	No
H	User name for collection	The username used by the collection method (scp or ftp) to log in to the remote server. If the collection method is cnrsdk, this field is not used.	No
I	Password for collection	The password used by the collection method (scp or ftp) to log in to the remote server. Used in conjunction with the 'User name for collection'. If the collection method is cnrsdk, this field is not used.	No
J	Collection port	The port number the collection method (scp or ftp) is listening on. If no value is specified, this will default to **22** if the collection method is scp, and **21** if the collection method is ftp. For CNR servers managed via the SDK, this will be the port that the CNR server is listening on for connections.	No
K	Container(s)	No longer used.	
L	VendorInfo	Vendor specific information for the product's collection type. Each item of information is specified in this single field by separating each field with the '|' character. For collection types **qip**, **adc**, **msft** and **isc**, the information includes the DHCP Configuration file pathname and DHCP Active Lease file pathname. For example, /opt/qip/dhcp/dhcpd.conf |/opt/qip/dhcp/dhcp.db or c:\qip\dhcp\dhcpd.conf |c:\qip\dhcp\dhcp.db For collection type **cnr** that are not managed via the SDK, the information includes the	No

Col	Field	Accepted Values	Required
		Path/Executable of NRCD command, the NRCMD user id, the NRCMD password and the Cluster Name. For example, /opt/cnr/bin/nrcmd|myuserid|mypass|cluster1 For CNR servers managed via the SDK, the directory component is not required, however the username and password to connect to the CNR machine are required as the second and third fields. For example, |myuserid|mypass	
M	WarningThreshold	Default scope utilization warning threshold. Provide warnings when usage of a pool assigned to this service is exceeded. If no value is specified, this will default to **90.**	No

ImportNetServiceWithTemplate

Overview

The **ImportNetServiceWithTemplate** CLI allows the user to bulk import DNS servers into IPAM by applying a pre-defined Server Template.

Using the IPAM GUI, create a DNS Server Template. This is accomplished through the System → Network Services Policies & Options → DNS Server Templates. Then use this CLI to create new servers using that template.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.ImportNetServiceWithTemplateCLI -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportNetServiceWithTemplate.sh -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportNetServiceWithTemplate.cmd -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the file to import. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-imported) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example imports network services with template from the *netserviceswithtemp.csv* file, places into the *netserviceswithtemp.reject* file any records that could not be imported, and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/ImportNetServiceWithTemplate.sh -u joe -p joepwd
-f netserviceswithtemp.csv -r netserviceswithtemp.reject -e importerrors.txt
```

File Format

Col	Field	Accepted Values	Required
A	Name	The name of the Network Service. This can be any combination of letters and numbers and should be a fully qualified domain name (FQDN).	Yes

Col	Field	Accepted Values	Required
B	IP Address	The IP address of the Network Service. This must be a valid IPv4 IP address.	Yes
C	Type	The type of Network Service. Accepted value is **dns**. If this column is left blank, **dns** is assumed.	No
D	Template name	The name of the DNS Server Template as defined in System → Network Services Policies & Options → DNS Server Templates. For example, **UNIX Standard for INS DNS**.	Yes
E	Agent name	The name of the IPAM Agent that is responsible for contacting this Network Service as defined in System → Agents.	Yes

ImportNetworkLink

Overview

The **ImportNetworkLink** CLI allows the user to bulk import logical network link definitions into IPAM. It also allows modification of existing logical and physical network links when using the overwrite option (-o), described later in this section.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.ImportNetworkLinkCLI -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-o] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportNetworkLink.sh -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-o] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportNetworkLink.cmd -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-o] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the file to import. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-imported) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.
-o	No	Overwrite option. Specify whether the import file contents will overwrite any matching records that exist in the database. See below.
-v	No	Produces verbose output.

Usage Example

This example imports network links from the *newlinks.csv* file, places into the *newlinks.reject* file any records that could not be imported, and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/ImportNetworkLink.sh -u joe -p joepwd -f newlinks.csv
-r newlinks.reject -e importerrors.txt
```

Import with overwite and the !BLANK! keyword

This example imports network links with the overwite option:

```
$INCHOME/etc/cli/ImportNetworkLink.sh -u joe -p joepwd -f networklinks.csv -o
```

Note the following:

- You can produce a file in the required format using the ExportNetworkLink CLI.
- The –o (overwite) parameter is required in order to modify network links via this CLI.
- The rows describe network links to be either imported or modified. In other words, existing network links will be modified; new network links will be imported. Network links to be modified are identified by network link name.
- The network link type (logical or physical) cannot be modified.
- To change which network link is assigned to a particular block in a logical container, use the ImportChildBlockCLI with the overwrite option.
- To clear a field in an existing record, specify "!BLANK!". An empty cell indicates no change will be made for that field. Only the following values may be cleared: description, DHCP Option Set, DHCP Policy Set.

File Format

Col	Field	Accepted Values	Required
A	Name	The name of the network link.	Yes
B	Type	The only value accepted on import is the default, **logical**. This column is provided for informational purposes for the ExportNetworkLink CLI. It will be ignored on overwrite.	No
C	Description	A description of the network link. Use "\n" to separate lines.	No
D	DHCP Option Set	The name of a DHCP Option Set defined within IPAM that will be applied to subnets using this network link.	No
E	DHCP Policy Set	The name of a DHCP Policy Set defined within IPAM that will be applied to subnets using this network link.	No
F	Block Names	A list of block names, separated by '|', that are assigned to this network link. This column is provided for informational purposes for the ExportNetworkLink CLI and is ignored on import and overwrite. To change the network link assigned to a block, use the ImportChildBlock CLI with the overwrite option.	No
G	Containers	A list of containers, separated by '|', that identify the containers for the corresponding blocks in column F. Only one container will be provided for each block to disambiguate blocks in overlapping address space. Like Block Names, this column is provided for informational purposes for the ExportNetworkLink CLI and is ignored on import and overwrite.	No

ImportPrefixPool

Overview

The **ImportPrefixPool** CLI allows the user to bulk import prefix pools into IPAM.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.ImportPrefixPoolCLI -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportPrefixPool.sh -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportPrefixPool.cmd -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the file to import. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-imported) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example imports prefix pools from the *newprefixpools.csv* file, places into the *newprefixpools.reject* file any records that could not be imported, and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/ImportPrefixPool.sh -u joe -p joepwd -f newprefixpools.csv
-r newprefixpools.reject -e importerrors.txt
```

File Format

Col	Field	Accepted Values	Required
A	Start Address	The IP Address of the first address in the pool. This address must be in a block with an In-Use/Deployed status.	Yes
B	Length	The length of the prefix pool.	Yes
C	Address Pool Type	One of "Dynamic PD DHCPv6" or "Automatic PD DHCPv6".	Yes
D	Name	Prefix Pool name. Defaults to "Start Address/Length"	No

Col	Field	Accepted Values	Required
E	Container	The name of the container that holds the block in which the pool is defined. This is required only if there is overlapping address space in use, and the start address is in overlapping space. The container is then used to uniquely determine the block that will contain the address pool.	No, unless Start address is not unique.
F	Delegated Prefix Length	The default length of the delegated prefix.	Yes
G	Longest Prefix Length	The longest prefix length allowed for delegated prefixes. CNR DHCP only.	No
H	Shortest Prefix Length	The shortest prefix length allowed for delegated prefixes. CNR DHCP only.	No
I	Primary Net Service	The name of the DHCP server that will serve addresses from this pool	No
J	DHCP Option Set	The name of an Option Set used with this pool.	No
K	DHCP Policy Set	The name of a Policy Set used with this pool.	No
L	Allow DHCP Client Classes	A list of Client Classes that are allowed in this address pool. Separate the list entries with a vertical bar "|". For example, to allow two client classes named "allowA" and "allowB", specify: allowA|allowB	No
M	Deny DHCP Client Classes	A list of Client Classes that are NOT allowed in this address pools. Separate the list entries with a vertical bar "|". For example, to disallow two client classes named "denyA" and "denyB", specify: denyA|denyB	No
N	OverlapInterfaceIp	Flag to allow a DHCPv6 prefix pool to overlap an interface address. This flag may be set only if pool is managed by a CNR DHCPv6 server.	No

ImportRootBlock

Overview

The **ImportRootBlock** CLI allows the user to bulk import root blocks into IPAM.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.ImportRootBlockCLI -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportRootBlock.sh -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportRootBlock.cmd -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the file to import. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-imported) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example imports Root Blocks from the *newrootblocks.csv* file, places into the *newrootblocks.reject* file any records that could not be imported, and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/ImportRootBlock.sh -u joe -p joepwd -f newrootblocks.csv
-r newrootblocks.reject -e importerrors.txt
```

File Format

Col	Field	Accepted Values	Required
A	Container	The name of the logical container that will hold the block. Names can be in either short or long format. Short format example: **Dallas**. Long format example: **InControl/Texas/Dallas**. Long format eliminates ambiguity in cases where there are duplicate container names.	Yes
B	IP space	The IP block to create. This should be in the format of a network address (e.g., 10.0.0.0).	Yes

Col	Field	Accepted Values	Required
C	Block size	The size of the block in short-notation (e.g., 24 for a 255.255.255.0 network).	Yes
D	Description	A description of the block. Use "\n" to separate lines.	No
E	Block type	The Block Type for the block. If not specified, a block type of **Any** is assumed.	No
F	Block Name	A name for the block. Defaults to system supplied name of *Address space/Block size*.	No
G	Allow Duplicate Space	Whether or not to allow duplicate (overlapping) address space in this block. Accepted values are **true** or **false**. If not specified, defaults to **false**.	No
H	Regional Internet Registry	The Regional Internet Registry this space was obtained from. Accepted values are: **Generic**, **RFC1918**, **ARIN**, **RIPE**, **APNIC**, **LACNIC**, and **AFRINIC**. If not specified, **Generic** is assumed.	No
I	Organization Id	The organization id for the Regional Internet Registry this space was obtained from. This id must be predefined in IPAM.	No
J	Allocation Reason	The name of a pre-existing Allocation Reason. If Allocation Reason is not currently in IPAM, this field is skipped.	No
K	Allocation Reason Description	A description of the reason for the allocation.	No
L	SWIP/Net Name	SWIP/Net name for the block.	Yes, if required by Container rules
M	Create Reverse Domains	Whether or not to automatically create reverse DNS domain(s) for this block. Accepted values are **true** or **false**. If not specified, defaults to **false**.	No
N	Domain Type	Specify the domain type for the reverse DNS domain(s) to be created when Create Reverse Domains is "true". If not specified, defaults to "Default".	No
O	User Defined Fields	A series of name=value pairs, where the name is the UDF field name/tag and the value is desired value. Multiple pairs can be specified by separating each pair with the '|' character. For example, **UDFone=value one |UDFtwo=value two.** If the UDF type is Checkbox, the valid values are "on" or "off". If the UDF type is Textarea, use "\n" to separate lines.	Yes, for UDFs defined as required fields.

ImportServiceSnapshot

Overview

The **`ImportServiceSnapshot`** CLI allows the user to import address pools discovered by either a "Collect DHCP Utilization" or "Global Synchronization of DHCP Servers" task. Typically, such tasks are used to query the current state of the network and perform difference analysis to compare actual deployment with the topology modeled in IPAM. However, during the initial setup of the system the queried information from these tasks can be used to populate the original IPAM model. This CLI then, is used to perform this initial population of discovered address pools.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.ImportServiceSnapshot -u <userId> -p <pswd>
-t <taskId> [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportServiceSnapshot.sh -u <userId> -p <pswd>
-t <taskId> [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportServiceSnapshot.cmd -u <userId> -p <pswd>
-t <taskId> [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-t <taskId>	Yes	The Id of the "Collect DHCP Utilization" or "Global Synchronization of DHCP Servers" task.

Usage Example

This example imports address pools discovered by task `12345`, places into the *servicesnapshot.reject* file any records that could not be imported, and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/ImportServiceSnapshot.sh -u joe -p joepwd -t 12345
```

ImportZone

Overview

The **ImportZone** CLI allows the user to bulk import DNS zones into IPAM, and to update existing DNS zones.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.ImportDnsZoneCLI -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportZone.sh -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportZone.cmd -u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the file to import. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-imported) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example imports zones from the *newzones.csv* file, places into the *newzones.reject* file any records that could not be imported, and reports errors to the *importerrors.txt* file.

Note: Zone options are not supported in the first release of this service.

```
$INCHOME/etc/cli/ImportZone.sh -u joe -p joepwd -f newzones.csv -r newzones.reject
-e importerrors.txt
```

CNR Servers

Important! For CNR servers, either Update Policies (Column R) or Allow Update (Column Q) MUST allow dynamic updates from *localhost* in order for IPAM to successfully manage the zone without having to overwrite the zone every time the resource records change.

File Format

Col	Field	Accepted Values	Required
A	Server	The network service name of the DNS Server.	Yes, if GalaxyName is not supplied.
B	Zone type	Specifies the type of zone being imported. Accepted values are **master**, **slave**, **forward** or **stub, static-stub, redirect**.	Yes
C	Domain Name	Domain name already defined to IPAM	Yes
D	Domain Type	Domain type name already defined to IPAM. If not specified, the "Default" domain type will be used.	No
E	Update Zone	Specify **true** to update an existing zone, **false** to import a new zone. If not specified, defaults to **false**. When **true**, specify all fields as you would for an import. Any field not specified on an update is cleared or set to its default value.	No
F	View	The name of the view in which the new zone will be created. If specified, the view must exist. If not specified, the new zone will be created in the view named 'Default.'	No
G	Filename	The name of the zone file that will be generated by IPAM. If not specified, will default to a system-generated value based on zone type.	No
H	Automatic Generation of NS/GLUE Records	Accepted values are **true** or **false**. If not specified, defaults to **true**.	No
I	MNAME	Specifies an alternative name for the primary or "master" DNS server that DDNS requests should be sent to for this zone. If not specified, no MNAME will be used. This field only applies to master zones. It will be ignored for all other zone types and any zone designated as an Alias zone.	No
J	Allow Zone Transfers to DNS Listener	Accepted values are **true** or **false**. If not specified, defaults to **true**. This field only applies to master zones. It will be ignored for all other zone types.	No
K	Masters	For zone types **slave** and **stub** only, specify the master server. This field is not valid for other zone types.	Yes for zone types **slave** and **stub**
L	Zone extensions Prior	Specifies the name of the file containing text lines that will be inserted prior to the resource records for this zone.	No
M	Zone extensions After	Specifies the name of the file containing text lines that will be inserted following the resource records for this zone.	No

Col	Field	Accepted Values	Required
N	Template Zone	Indicates if the imported zone is intended to be a template zone. Accepted values are true or false. If not specified, defaults to false.	No
O	Alias Zone	Indicates if the imported zone is intended to be a alias zone. The linked template zone will be chosen by using the zone assigned to the template domain to which the alias domain is linked. Accepted values are true or false. If not specified, defaults to false.	No
P	Galaxy Name	Future Use	No
Q	Allow Update ACL	Specifies the DNS 'allow-update' address match list a BIND based server uses to filter DDNS update requests against. This free text field only applies to master zone types with dynamic updates enabled and is accepted without input validation. Setting this field value to "!BLANK!" on a zone update will serve to completely remove the 'allow-update' option from the zone. This field is ignored for non-master zones. This field is not supported by Microsoft DNS servers.	Yes, when Dynamic Zone is true, one of Allow Update ACL or Update Policy must be specified.
R	Update Policy	Specifies an update policy, already defined in IPAM, for a dynamic master zone. Setting this field value to "!BLANK!" on a zone update will serve to completely remove the update policy option from the zone. This field is ignored for non-master and non-dynamic zones. This field is not supported by Microsoft DNS servers.	See Allow Update ACL
S	Dynamic Zone	Accepted values are **true** or **false**. If not specified, defaults to **false**. True indicates that this is a dynamic zone. Applicable only for master zones and BIND based servers. When **true**, you must specify either Update Policy or Allow Update ACL. For Microsoft DNS servers, this defaults to **true**, and Update Policy and Allow Update ACL are not supported.	No
T	Publish NS	Accepted values are **true** or **false.** If not specified defaults to **true.** True indicates that this server will have an NS record published in the zone file. This corresponds to the "Include this server in NS record list" checkbox on the zone profile.	No

ImportZoneResourceRecord

Overview

The **ImportZoneResourceRecord** CLI allows the user to bulk import DNS resource records for a zone into IPAM.

Note: this interface should not be confused with the **ImportDomainResourceRecord** CLI, which is used to add records to a **domain**. **ImportZoneResourceRecord** is only effective when the 'Automatic Generation of NS/Glue Records' is set to OFF on the target zone.

Usage

Direct

```
$INCHOME/jre/bin/java –cp $CLASSPATH
com.diamondip.ipcontrol.cli.ImportZoneResourceRecordCLI –u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportZoneResourceRecord.sh –u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ImportZoneResourceRecord.cmd –u <userId> -p <pswd>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the file to import. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-imported) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example imports resource records from the *newresourcerecs.csv* file, places into the *newresourcerecs.reject* file any records that could not be imported, and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/ImportZoneResourceRecord.sh –u joe –p joepwd –f newresourcerecs.csv
–r newresroucerecs.reject –e importerrors.txt
```

File Format

Col	Field	Accepted Values	Required
A	Server	The network service name of the DNS Server.	Yes
B	View	The name of the view for this zone. If supplied the view must exist. If not specified, 'Default' will be used.	Yes
C	Zone	The name of the zone, which is the top level domain name.	Yes
D	Owner	The OWNER section of the resource record. Note that this section is specific to the type of resource record. Refer to the appropriate RFC for exact text that should be entered.	Yes
E	TTL	The Time To Live.	No
F	Class	The value currently supported is **IN**. If not specified, defaults to **IN**.	No
G	Resource Record Type	The type of resource record being imported. Accepted values are **A** and **NS**.	Yes
H	Data	The text for the DATA area of the resource record. Note that this section is specific to the type of resource record. Refer to the appropriate RFC for exact text that should be entered.	Yes

RestoreDeletedDevice

Overview

The **RestoreDeletedDevice** CLI restores deleted devices in the system.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.RestoreDeletedDeviceCLI -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/RestoreDeletedDevice.sh -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/RestoreDeletedDevice.cmd -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	Username
-p <pswd>	Yes	Password
-?	No	Print help
-f <update filename>	Yes	The name of the CSV file containing modify instructions. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

```
$INCHOME/etc/cli/RestoreDeletedDevice.sh -u joe -p joepwd -f restoreDeletedDevs.csv -r
restoreDeletedDevs.reject -e restoreDeletedDevs.err
```

Output

If successful, the CLI restores the deleted devices per the input file and exits.

File Format

The input to this CLI is the output of **ExportDeviceRestoreList** CLI. ExportDeviceRestoreList CLI may be run using suitable options to find the records, examine the generated file, remove records that do not need to be restored and use the file as input to this CLI.

Col	Field	Accepted Values	Required
A	Restore id	The id used internally to identify the record. It is required to restore this record via RestoreDeletedDevice.	Yes
B	Date/Time	The date and time when the device was deleted.	No
C	Administrator	The login id of the administrator who deleted the record.	No
D	IP Address	The ipaddress of the deleted device.	No
E	Hostname	The hostname of the deleted device.	No
F	Block name	The name of the block that the device was deleted from.	No
G	Container	Name of the Container from which the device was deleted.	No
H	Device Type	Device type of the deleted device	No
I	Address Type	Address type of the deleted device.	No
J	Description	Description of the deleted device.	No
K	DUID	DHCP Unique Identifier	No
L	Hardware type	Hardware type of the deleted device. Ethernet or Token Ring	No
M	Hardware Address	The Hardware(Mac)Address of the deleted device	No
N	Domain Name	Domain name of the deleted device.	No
O	Domain Type	Domain type of the deleted device.	No
P	Ignore Duplicate Warning flag	Set to True to ignore Duplicate Warnings for mac address or hostname while restoring. Defaults to false.	No

RestoreDeletedResourceRecords

Overview

The **RestoreDeletedResourceRecords** CLI restores deleted resource records in the system.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.RestoreDeletedResourceRecordsCLI -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/RestoreDeletedResourceRecords.sh -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/RestoreDeletedResourceRecords.cmd -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	Username
-p <pswd>	Yes	Password
-?	No	Print help
-f <update filename>	Yes	The name of the CSV file containing modify instructions. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

```
$INCHOME/etc/cli/RestoreDeletedResourceRecords.sh -u joe -p joepwd -f restoreDeletedRRs.txt
-r restoreDeletedRRs.reject -e restoreDeletedRRs.err
```

Output

If successful, the CLI restores the deleted resource records per the input file and exits.

File Format

The input to this CLI is the output of **ExportResourceRecordRestoreList** CLI. ExportResourceRecordRestoreList CLI may be run using suitable options to find the records, examine the generated file, remove records that do not need to be restored and use the file as input to this CLI.

Col	Field	Accepted Values	Required
A	Restore id	The id used internally to identify the record. It is required to restore this record via RestoreDeletedResourceRecord.	Yes
B	Date/Time	The date and time when the record was deleted.	No
C	Administrator	The login id of the administrator who deleted the record.	No
D	Domain	The name of the domain to which the resource records belongs.	No
E	Domain Type	The name of the domain type to which the domain belongs.	No
F	Owner	The OWNER section of the resource record. Note that this section is specific to the type of resource record. Refer to the appropriate RFC for exact text format.	No
G	TTL	The Time To Live.	No
H	Class	The value currently supported is **IN**.	No
I	Resource Record Type	The type of resource record.	No
J	Data	The text for the DATA area of the resource record. Note that this section is specific to the type of resource record. Refer to the appropriate RFC for exact text format.	No
K	Comment	Text appended to the resource record.	No
L	Hostname	Hostname of the device from which the resource record was deleted. Filled only if a device resource record was deleted.	No
M	IP Address	IP Address of the device from which the resource record was deleted. Filled only if a device resource record was deleted.	No
N	Ignore Duplicate Warning flag	Set to True to ignore Duplicate Warnings for Owner field.	No

Exports

Export CLIs allow you to retrieve and filter information from IPAM. Data may be viewed on the screen, or output into a file suitable for modifying and importing.

Note that user access control lists (ACL) are enforced, which may affect export results. Users are not allowed to perform exports on data without the appropriate Read rights in the system.

ExportAdmin

Overview

The **ExportAdmin** CLI exports a list of administrators into a specified file. This file can be modified and then imported using the **ImportAdmin** CLI.

Usage

Direct

```
$INCHOME/jre/bin/java –cp $CLASSPATH
com.diamondip.ipcontrol.cli.ExportAdminCLI –u <userId>  -p <pswd>
[-at <admin type>] [-id <login id>] [-r <role>] [-fn <first name>]
[-ln <last name>][-f <export-file>] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ ExportAdmin.sh –u <userId>  -p <pswd>
[-at <admin type>] [-id <login id>] [-r <role>] [-fn <first name>]
[-ln <last name>] [-f <export-file>] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ ExportAdmin.cmd –u <userId>  -p <pswd>
[-at <admin type>] [-id <login id>] [-r <role>] [-fn <first name>]
[-ln <last name>] [-f <export-file>] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <export filename>	No	The name of the file to export the data to. If no file name is specified, results are outputted to the screen.
-at <admin type>	No	Filter. Allows you to filter export based on the administrator type. Specify **master, normal** or **readonly**.
-id <login id>	No	Filter. Specify the login id of a specific administrator or use the wildcard character '*' to perform a partial match on the id. If the login name contains embedded spaces, surround the name with double-quotes " ".
-r <role name>	No	Filter. Allows you to filter export based on the administrator's roles. Specify the role name or use the wildcard character '*' to perform a partial match on the role name. If the name contains embedded spaces, surround the name with double-quotes " ".
-ln <last name>	No	Filter. Specify the last name of the administrator or use the wildcard character '*' to perform a partial match on the name. If the name contains embedded spaces, surround the name with double-quotes " ".
-fn <first name>	No	Filter. Specify the first name of the administrator or use the wildcard character '*' to perform a partial match on the name. If the name contains embedded spaces, surround the name with double-quotes " ".
-v	No	Produces verbose output.

Usage Example

This example exports all administrators to the file *adexport.csv* in the current directory.

```
$INCHOME/etc/cli/ExportAdmin.sh -u joe -p joepwd -f adexport.csv
```

This example exports administrators of type **MASTER** to the file *adexport.csv*.

```
$INCHOME/etc/cli/ExportAdmin.sh -u joe -p joepwd -f adexport.csv -at master
```

File Format

This CLI uses the same format that the **ImportAdmin** CLI requires.

Col	Field	Accepted Values
A	Login ID	Administrator login ID (for example, jsmith10)
B	Login Password	Administrator's password. On export, this is always shown as "********".
C	First Name	First (given) name of administrator
D	Last Name	Last (family) name of administrator
E	Address1	Mailing address
F	Address2	Additional line for mailing address
G	Address3	Additional line for mailing address
H	Email	Email address
I	Phone	Phone number
J	Pager	Paging number
K	Fax	Fax number
L	Authorize Externally	Determines whether this user's username and password will be passed to an external authentication system. This is setup by the IPAM administrator in the System Policies screen. Accepted values are **true** or **false**.
M	Enabled	Determines whether this user can access the IPAM system. Accepted values are **true** or **false**.
N	Admin Type	One of **MASTER**, **NORMAL**, or **READONLY**. **MASTER** users have full control over the entire IPAM system. **NORMAL** users have ordinary read-write permissions, and may be restricted to working with only certain portions of the IPAM system. **READONLY** users have read-only access to IPAM, and may be restricted to seeing only certain portions of the IPAM system.
O	Role	Administrator Roles for this administrator. Multiple roles are separated by '|'.
P	Assignable Role	Assignable Administrator Roles for this administrator. Assignable Roles are roles that the Administrator can assign to another Administrator. Multiple roles are separated by '|'.

Col	Field	Accepted Values
Q	Authorized Functions	**ALL**, **NONE**, or a list of functions and main headings the administrator can access, separated by '|'. Each function corresponds to a menu item that this administrator is allowed to access. List can include a main heading like **IPAM**. List can include individual functions like **AddSites**. A complete list of keywords for the functions and headings is provided in the ImportAdmin section of this guide. These keywords are case-insensitive.
R	Container Access Control	**ALL**, **NONE**, or a list of containers with access indication. Multiple containers are separated by '|'. Access indicators are separated by '^'. The access indicators are specified as: Name^read^write^delete^apply to children^ *device approve access For example: InControl/US/ East^Y^N^N^Y^Y|West^Y^Y^Y^Y^Y **This will be exported regardless of the workflow policy setting*
S	Block Access Control	A list of blocks with access indication. This is used to fine-tune the blocks given access by the container access control rules. You typically add blocks to an administrator role ACL when you want to override the privileges defined on the container for a specific block. Therefore, when this column is blank, the block access is defined by the container access control rules. Multiple blocks are separated by '|'. Access indicators are separated by '^'. The access indicators are specified as: Block name^container name^block status^read^write^delete^ *device approve access For example: 10.0.0.0/24^InControl/US/East^deployed^Y^N^N^Y | 2001::4:0/110^West^aggregate^Y^Y^Y^Y ** This will be exported regardless of the workflow policy setting*
T	Block Type Access	**ALL**, **NONE**, or a list of allowed block types. For example: Any|Data|VoIP To limit allowed block sizes, the allowed sizes, or **NONE**, will be shown following the block type, separated by '/'. Any/*V4 specification*/*V6 specification* | Data/*V4 specification*/*V6 specification* where *V4* and *V6 specification* are a list of ^ separated block sizes: Any/24^25|Data/22^23/65^66|VoIP//67^68 In the above example, Any is limiting block sizes for V4 only, and VoIP is limiting block sizes for V6 only. Any/24^25|Data/22^23/65^66|VoIP/NONE/67^68 In the above example, VoIP is limiting block sizes to allow only 2 V6 block sizes, and no V4 block sizes are allowed.
U	Device Type Access	**ALL**, **NONE**, or a list of allowed device types. For example: PC|Server|Switch

Col	Field	Accepted Values
V	Policies	Policy selections are specified in the following order, separated by '/': CLI Access/duplicate hostname checking/checking style/allow dup A recs/allow dup hardware addr CLI Access: **true** or **false** Dup hostname checking: **I**(gnore), **W**(arn) or **F**(ail) Dup hostname checking style: **F** (fully qualified) or **H** (hostname only) Allow duplicate A records: **I**(gnore), **W**(arn) or **F**(ail) Allow duplicate hardware addresses: **I**(gnore), **W**(arn) or **F**(ail) For example: false/I/H/W/F
W	Domain Access Control	**ALL**, **NONE**, or a list of domains with access indication. Multiple domains are separated by '|'. Access indicators are separated by '^'. The access indicators are specified as: Domain name/domain type ^read^write^delete^RR access^RR write^apply to children^*Resource Record Approve Access For example: sample.com/default^Y^N^N^N^N^N^Y ** This will be exported regardless of the workflow policy setting*
X	Net Service Access Control	**ALL**, **NONE**, or a list of servers with access indication. Multiple services are separated by '|'. Access indicators are separated by '^'. The access indicators are specified as: Name/type^read^write^deploy Type is either **DNS** or **DHCP**. For example: DnsOne/DNS^Y^N^N
Y	Resource Record Type Access Control	**ALL**, **NONE**, or a list of allowed resource record types. For example: A|PTR|CNAME *Note: "NSAP-PTR" is exported as **NSAPPTR** and "Zone RR" is exported as **ZONERR**.*
Z	Address Type Access	**ALL**, **NONE**, or a list of allowed address types. For example: STATIC|RESERVED

ExportAdminRole

Overview

The **ExportAdminRole** CLI exports a list of administrator roles into a specified file. This file can be modified and then imported using the **ImportAdminRole** CLI.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.ExportAdminRoleCLI -u <userId>  -p <pswd>
[-n <role-name>] [-de <description>] [-f <export-file>] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ ExportAdminRole.sh -u <userId>  -p <pswd>
[-n <role-name>] [-de <description>] [-f <export-file>] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ ExportAdminRole.cmd -u <userId>  -p <pswd>
[-n <role-name>] [-de <description>] [-f <export-file>] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <export filename>	No	The name of the file to export the data to. If no file name is specified, results are outputted to the screen.
-n <role name>	No	Filter. Allows you to filter export based on the role name. Specify the role name or use the wildcard character '*' to perform a partial match on the role name. Surround the parameter with double quotes if there is an embedded blank, for example, -r "admin role one".
-de <description>	No	Filter. Allows you to filter export based on the administrator role description or use the wildcard character '*' to perform a partial match on the description. Surround the parameter with double quotes if there is an embedded blank, for example, -de "two words".

Usage Example

This example exports all administrator roles to the file *arexport.csv* in the current directory.

```
$INCHOME/etc/cli/ExportAdminRole.sh -u joe -p joepwd -f arexport.csv
```

This example exports administrator roles where the description contains the word "HR" to the file *arexport.csv*.

```
$INCHOME/etc/cli/ExportAdminRole.sh -u joe -p joepwd -f arexport.csv -de "*HR*"
```

File Format

This CLI uses the same format that the **`ImportAdminRole`** CLI requires.

Col	Field	Accepted Values
A	Role Name	Administrator role name.
B	Description	Administrator role description.
C	Authorized Functions	**ALL**, **NONE**, or a list of functions and main headings the administrator can access, separated by '|'. Each function corresponds to a menu item that this administrator is allowed to access. List can include a main heading like **IPAM**. List can include individual functions like **AddSites**. A complete list of keywords for the functions and headings is provided in the ImportAdmin section of this guide. These keywords are case-insensitive.
D	Container Access Control	**ALL**, **NONE**, or a list of containers with access indication. Multiple containers are separated by '|'. Access indicators are separated by '^'. The access indicators are specified as: Name^read^write^delete^apply to children^ *device approve access For example: InControl/US/ East^Y^N^N^Y^Y|West^Y^Y^Y^Y^Y **This will be exported regardless of the workflow policy setting*
E	Block Access Control	A list of blocks with access indication. This is used to fine-tune the blocks given access by the container access control rules. You typically add blocks to an administrator role ACL when you want to override the privileges defined on the container for a specific block. Therefore, when this column is blank, the block access is defined by the container access control rules. Multiple blocks are separated by '|'. Access indicators are separated by '^'. The access indicators are specified as: Block name^container name^block status^read^write^delete^ *device approve access For example: 10.0.0.0/24^InControl/US/East^aggregate^Y^N^N^Y | 2001::4:0/110^West^deployed^Y^Y^Y^Y ** This will be exported regardless of the workflow policy setting*

Col	Field	Accepted Values
F	Block Type Access	**ALL**, **NONE**, or a list of allowed block types. For example: Any|Data|VoIP To limit allowed block sizes, the allowed sizes, or **NONE**, will be shown following the block type, separated by '/'. Any/*V4 specification/V6 specification* | Data/*V4 specification/V6 specification* where *V4* and *V6 specification* are a list of ^ separated block sizes: Any/24^25|Data/22^23/65^66|VoIP//67^68 In the above example, Any is limiting block sizes for V4 only, and VoIP is limiting block sizes for V6 only. Any/24^25|Data/22^23/65^66|VoIP/NONE/67^68 In the above example, VoIP is limiting block sizes to allow only 2 V6 block sizes, and no V4 block sizes are allowed.
G	Device Type Access	**ALL**, **NONE**, or a list of allowed device types. For example: PC|Server|Switch
H	Policies	Policy selections are specified in the following order, separated by '/': CLI Access/duplicate hostname checking/checking style/allow dup A recs/allow dup hardware addr CLI Access: **true** or **false** Dup hostname checking: **I**(gnore), **W**(arn) or **F**(ail) Dup hostname checking style: **F** (fully qualified) or **H** (hostname only) Allow duplicate A records: **I**(gnore), **W**(arn) or **F**(ail) Allow duplicate hardware addresses: **I**(gnore), **W**(arn) or **F**(ail) For example: false/I/H/W/F
I	Domain Access Control	**ALL**, **NONE**, or a list of domains with access indication. Multiple domains are separated by '|'. Access indicators are separated by '^'. The access indicators are specified as: Domain name/domain type ^read^write^delete^RR access^RR write^apply to children^*Resource Record Approve Access For example: sample.com/default^Y^N^N^N^N^N^Y ** This will be exported regardless of the workflow policy setting*
J	Net Service Access Control	**ALL**, **NONE**, or a list of servers with access indication. Multiple services are separated by '|'. Access indicators are separated by '^'. The access indicators are specified as: Name/type^read^write^deploy Type is either **DNS** or **DHCP**. For example: DnsOne/DNS^Y^N^N
K	Resource Record Type Access Control	**ALL**, **NONE**, or a list of allowed resource record types. For example: A|PTR|CNAME *Note: "NSAP-PTR" is exported as **NSAPPTR** and "Zone RR" is exported as **ZONERR**.*
L	Address Type Access	**ALL**, **NONE**, or a list of allowed address types. For example: STATIC|RESERVED

ExportChildBlock

Overview

The **ExportChildBlock** CLI exports a list of all the Child Blocks into a specified file. This file can be modified and then imported using the **ImportChildBlock** CLI.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.ExportChildBlockCLI -u <userId>  -p <pswd>
[-n name] [-i ipaddress>] [-j <interface>] [-d <user defined field name=value>]
[-r <ipaddressrange>] [-st <status>] [-b <block>] [-c <container>] [-t <blocktype>]
[-f <export-file>] [-pb <parentBlock>] [-pc <parentContainer>] [-iv <version>] [-h] [-l]
[-ex] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ExportChildBlock.sh -u <userId>  -p <pswd>
[-n name] [-i <ipaddress>] [-j <interface>] [-d <user defined field name=value>]
[-r <ipaddressrange>] [-st <status>] [-b <block>] [-c <container>] [-t <blocktype>]
[-f <export-file>] [-pb <parentBlock>] [-pc <parentContainer>] [-iv <version>] [-h] [-l]
[-ex] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ExportChildBlock.cmd -u <userId>  -p <pswd>
[-n name] [-i <ipaddress>] [-j <interface>] [-d <user defined field name=value>]
[-r <ipaddressrange>] [-st <status>] [-b <block>] [-c <container>] [-t <blocktype>]
[-f <export-file>] [-pb <parentBlock>] [-pc <parentContainer>] [-iv <version>]  [-h] [-l]
[-ex] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <export filename>	No	The name of the file to export the data to. If no file name is specified, results are outputted to the screen.
-b <block address/size>	No	Filter. Specify the CIDR notation of a block. That is, specify the starting IP address for the block, followed by the block size, separated by a slash '/'. For example, 10.0.0.0/24. The wildcard character '*' can be used in the start address only.
-c <container name>	No	Filter. Allows you to filter export based on the name of the container where the block is located. You may use a fully qualified container name or partial container name with the '*' wildcard. A combination of fully qualified container name and '*' wildcard is not supported.
-d <user defined field name=value>	No	Filter. Specify a User Defined Field (UDF) attached to a block. The UDF is specified using the syntax name=value, where name is the name of the UDF, and value is the value of the UDF. For wildcarding, use a "*" character within "double quotes", i.e., –d "udfName=value*". Also surround the parameter with double quotes if there is an embedded blank in the filter string. Note that filtering by UDF of value data type URL is not supported.

Parameter	Required	Description
-de <description>	No	Filter. Allows you to filter export based on the block description. Surround the parameter with double quotes if there is an embedded blank, for example, -de "two words".
-ex	No	Expanded mode. Additional columns will be appended for each UDF defined in the exported records. The first row of the file will be a comment header row, beginning with "^", with each column's field name, including the tag name for each UDF. The architected UDF column P will contain "Expanded". See below for an example.
-h (--recurse) <recurse option>	No	Filter. Allows you to provide a recurse option for the export. Without arguments: Only valid when –pb is specified. When present, recursively exports child blocks of the named parent block. Option **container**: Only valid when –c is specified. When present, recursively exports child blocks from the container branch of the container tree (named via the –c option) and all its descendants. See below for an example.
-i <IP address>	No	Filter. Specify an IP address which falls within the start and end address of a block.
-iv <IP version>	No	Filter. Specify an IP address version. Specify **V4** or **V6**.
-j <Interface>	No	Filter. Specify the name of an interface to which the block must be attached.
-l (--includeFreeBlocks)	No	Flag. By default, the list of exported child blocks will not include the free blocks which are maintained by IPAM. Include this Boolean option for the free blocks to be included in the export.
-n <block name>	No	Filter. Specify the name of a specific block or use the wildcard character '*' to perform a partial match on the name. If the block name contains embedded spaces, surround the name with double-quotes " ".
-pb < parentBlock >	No	Filter. Specify the name of the specific block's parent or starting address followed by /CIDR size
-pc <parentContainer>	No	Filter. Specify the name of the specific block's parent block's container. Only valid when accompanied by –pb option. Specifying the container can help to avoid ambiguity. A fully qualified container name may be supplied.
-r <IP address range>	No	Filter. Specify a range of IP addresses which span one or more blocks. The format of the range is specified as start-end. For example, 10.0.0.0-10.0.255.255.
-st <block status>	No	Filter. Specify the block status. Accepted values are: **free**, **aggregate**, **reserved**, **subnet**, **fullyassigned**. Note: if blocks of status **free** are desired, the –l option must be used.
-t <block type>	No	Filter. Specify the name of a specific block type or use the wildcard character '*' to perform a partial match on the block type name.

Usage Example

This example exports all child blocks to the file *rbexport.csv* in the current directory.

```
$INCHOME/etc/cli/ExportChildBlock.sh –u joe –p joepwd –f rbexport.csv
```

This example exports child blocks that begin with the name `MyBlock` and that are of block type `Private` to the file *rbexport.csv*.

```
$INCHOME/etc/cli/ExportChildBlock.sh –u joe –p joepwd –f rbexport.csv –n MyBlock*
–t Private
```

This example exports child blocks from the container named `MyContainer` and all its descendants to the file *rbexport.csv*.

```
$INCHOME/etc/cli/ExportChildBlock.sh –u joe –p joepwd –f rbexport.csv –c MyContainer
–h container
```

This example exports IPv6 child blocks from the container named `MyContainer` and all its descendants to the file *rbexport.csv*.

```
$INCHOME/etc/cli/ExportChildBlock.sh –u joe –p joepwd –f rbexport.csv –c MyContainer
–h container –iv V6
```

Export via parent block name parameter example

This example exports all child blocks of the parent block named `172.16.0.0/23`.

```
$INCHOME/etc/cli/ExportChildBlock.sh –u joe –p joepwd –f rbexport.csv –pb 172.16.0.0/23
```

This example exports all child blocks of the parent block named `172.16.0.0/23` **recursively**.

```
$INCHOME/etc/cli/ExportChildBlock.sh –u joe –p joepwd –f rbexport.csv –pb 172.16.0.0/23 –h
```

This example exports all child blocks of the parent block named `172.16.0.0/23` from the container named `MyContainer` and all its descendants.

```
$INCHOME/etc/cli/ExportChildBlock.sh –u joe –p joepwd –f rbexport.csv –pb 172.16.0.0/23 –c
MyContainer –h container
```

This example exports all child blocks of the parent block named `172.16.0.0/23` contained in container /InControl/Canada/North.

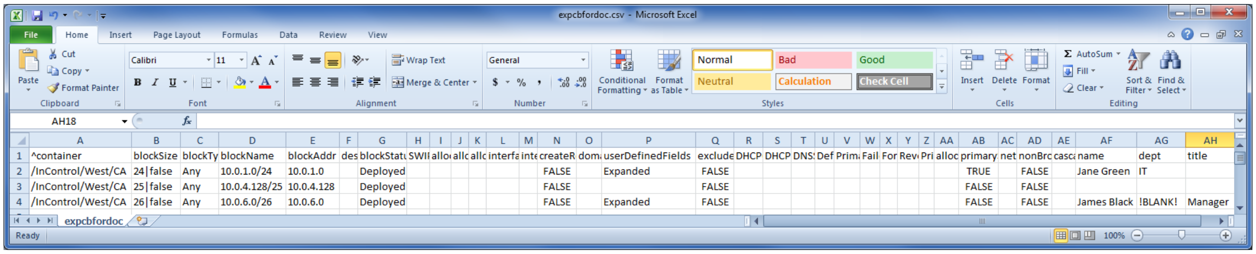
```
$INCHOME/etc/cli/ExportChildBlock.sh –u joe –p joepwd –f rbexport.csv –pb “172.16.0.0/23”
–pc “/InControl/Canada/North”
```

Export using expanded format example

This example exports all child blocks in the container named CA, with the output in expanded format.

```
$INCHOME/etc/cli/ExportChildBlock.sh -u joe -p joepwd -f cexport.csv -c CA -ex
```

Here is the output file:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH
1	^container	blockSize	blockTy	blockName	blockAddr	des	blockStatu	SWII	allo	allc	allc	interfa	inte	createR	dom	userDefinedFields	exclude	DHCP	DHCP	DNS	Def	Prim	Fail	For	Rev	Pri	alloc	primary	net	nonBrc	casc	name	dept	title
2	/InControl/West/CA	24|false	Any	10.0.1.0/24	10.0.1.0		Deployed							FALSE		Expanded	FALSE											TRUE		FALSE		Jane Green	IT	
3	/InControl/West/CA	25|false	Any	10.0.4.128/25	10.0.4.128		Deployed							FALSE			FALSE											FALSE		FALSE				
4	/InControl/West/CA	26|false	Any	10.0.6.0/26	10.0.6.0		Deployed							FALSE		Expanded	FALSE											FALSE		FALSE		James Black	!BLANK!	Manager

Note the following:

- The first row is a header line, beginning with "^". The column headers beyond the architected column AC contain the user defined field tags.
- Column P contains "Expanded" when the row includes user defined fields beyond column AE. Row 3 has no user defined fields defined for that block.
- Only those user defined field tags defined for the blocks exported (based on container and blockType) will have columns in the output file.
- In this example, columns AF-AH contain the values for the user defined fields. If there is no value, as in row 2 column AH, the cell will be blank.
- You can use ImportChildBlock to import an expanded format exported file by using the –o (overwrite) parameter.

File Format

The format for the export file is as follows.

Note: This is the same format that the **`ImportChildBlock`** CLI requires.

Col	Field	Accepted Values	Required
A	Container	The name of the container that will hold the block. Names can be in either short or long format. Short format example: **Dallas**. Long format example: **InControl/Texas/Dallas**. Long format eliminates ambiguity in cases where there are duplicate container names.	Yes
B	Block size | IPV6	The size of the block in short-notation (e.g., 24 for a 255.255.255.0 network). If an IPV6 block is exported, the block size will be followed by "|true", for IPV4 "|false".	Yes | No
C	Block type	The Block Type for the block If not specified, a block type of **Any** is assumed.	No

Col	Field	Accepted Values	Required
D	Block Name	A name for the block. Defaults to system supplied name of *Address space/Block size*.	No
E	Address Block	The address block to allocate. If no address block is specified, space will be auto-allocated.	No
F	Description	A description of the block.	No
G	Current Status	The current status of the block. Accepted values are: **Deployed**, **FullyAssigned**, **Reserved**, **Aggregate**.	Yes
H	SWIP Name	SWIP name for the block.	Yes, if required by Container rules
I	Allocation Reason	The name of a pre-existing Allocation Reason. If Allocation Reason is not currently in IPAM, this field is skipped.	No
J	Allocation Reason Description	A description of the reason for the allocation. Wrap the statement in "quotes" if it contains any commas.	No
K	Allocation Template	If this block is being added to a device container with blockStatus=**Deployed**, the name of the allocation template to use to create address pools from the newly created block.	No
L	Interface Name	If this block is being added to a device container, the name of the interface to attach the block to.	Yes, if block is being added to device container. Otherwise, no.
M	Interface Offset or Address	The specific address(es) for the interface IP address(s). Multiple addresses will be separated by the '|' character. For example, 10.0.0.1|10.0.0.2|10.0.0.3	No. An offset of 1 is assumed if none is specified.
N	Create Reverse Domains	Whether or not to automatically create reverse DNS domain(s) for this block. Accepted values are **true** or **false**. If not specified, defaults to **false**.	No
O	Domain Type	Specify the domain type for the reverse DNS domain(s) to be created when Create Reverse Domains is "true". If not specified, defaults to "Default".	No
P	User Defined Fields	A series of name=value pairs, where the name is the UDF name and the value is desired value. Multiple pairs can be specified by separating each pair with the '|' character. For example, **UDFone=value one |UDFtwo=value two.** If the UDF type is Checkbox, the valid values are "on" or "off". When this column contains the word "Expanded", the –ex option was invoked. See below for an example of the results.	Yes, for UDFs defined as required fields.

Col	Field	Accepted Values	Required
Q	Exclude From Discovery	Flag indicating if this subnet should be included in Host Discovery tasks. Accepted values are **true** or **false**. If not specified, defaults to **false**. Valid only for Deployed blocks.	No
R	DHCP Option Set	The name of a DHCP Option Set defined within IPAM that should apply to this subnet. **Valid only for Deployed blocks.**	No
S	DHCP Policy Set	The name of a DHCP Policy Set defined within IPAM that should apply to this subnet. **Valid only for Deployed blocks.**	No
T	DNS Servers	The list of default DNS Servers for this subnet. This list is supplied to DHCP clients on this subnet. The server name or IP Address is valid. For multiple servers, separate the server names with a vertical bar (|). **Valid only for Deployed blocks.**	No
U	Default Gateway	The default gateway address for this subnet. This address is supplied to DHCP clients on this subnet. **Valid only for Deployed blocks.**	No
V	Primary DHCP Server	The name or IP Address of the primary DHCP server for this address space. **Valid only for Deployed blocks.**	No
W	Failover DHCP Server	The name or IP Address of the failover DHCP Server for this address space. **Valid only for Deployed blocks.**	No
X	DNS Forward Domains	The list of DNS Forward domains for this address space, separated by a vertical bar (|). This list will appear in the GUI when choosing domains for devices. To specify a domain type, specify the domain followed by '/' followed by the domain type. **Valid only for Deployed blocks.** For example: hr.ins.com.|dmz.com./External In this example, hr.ins.com uses the default domain type, and dmz.com is of type 'External'.	No
Y	DNS Reverse Domains	The list of DNS Reverse domains for this address space, separated by a vertical bar (|). This list will appear in the GUI when choosing domains for devices. To specify a domain type, specify the domain followed by '/' followed by the domain type. **Valid only for Deployed blocks.** For example: 0-15.1.0.10.in-addr.arpa. /External |40.10.in-addr.arpa. In this example, 0-15.1.0.10.in-addr.arpa. is of type 'External', and 40.0.10.in-addr.arpa. uses the default domain type.	No
Z	Primary WINS Server	The IP Address of the Primary WINS Server for this subnet. Used to provide this information to DHCP for Dynamic Address types. Multiple WINS servers may be specified, separated by a comma.	No

Col	Field	Accepted Values	Required
AA	Allocation Strategy	Exported with no value as a place holder for consistency with ImportChildBlock.	No
AB	Primary Subnet	Flag indicating if this subnet is primary. Accepted values are **true** or **false**. **Valid only for Deployed blocks in device containers.**	No
AC	Network Link	The name of the Shared Network Segment for this subnet.	No
AD	NonBroadcast	Flag indicating if this is a Non-Broadcast subnet. Non-broadcast subnets are allowed to assign devices to the subnet and broadcast addresses. Accepted values are **true** or **false**. **Valid only for IPv4 Deployed block.**	No
AE	CascadePrimaryDHCP Server	Import with overwrite only	No

ExportContainer

Overview

The **ExportContainer** CLI exports containers into a specified file. This file can be modified and then imported using the **ImportContainer** CLI.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.ExportContainerCLI -u <userId> -p <pswd>
[-f <export filename>] [-n containerName] [-d udfName=value] [-fp] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ExportContainer.sh -u <userId> -p <pswd>
[-f <export filename>] [-n containerName] [-d udfName=value] [-fp] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ExportContainer.cmd -u <userId> -p <pswd>
[-f <export filename>] [-n containerName] [-d udfName=value] [-fp] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <export filename>	No	The name of the file to export the data to. If omitted, results are sent to the screen.
-n [containerName]	No	Filter. Allows you to filter export based on the name of the container. Use a "*" character (within 'single quotes', i.e., –n 'ContainerName*') for wildcarding. NOTE: if both –n and –d are specified, the export includes containers that satisfy *both* criteria.
-d [udfName=value]	No	Filter. Allows you to filter export based the user defined field name and value. Only containers that have this UDF set to this value will be exported. For wildcarding, use a "*" character within "double quotes", i.e., –d "udfName=value*". Also surround the parameter with single quotes if there is an embedded blank in the filter string. NOTE: if both –n and –d are specified, the export includes containers that satisfy *both* criteria. Also note that filtering by UDF of value data type URL is not supported.
-fp	No	Populates the parent container field using the long format, for example: InControl/Texas/Dallas
-v	No	Produces verbose output.

Usage Examples

This example exports all containers to the file named *exportcontainer.csv* file in the current directory.

```
$INCHOME/etc/cli/ExportContainer.sh -u joe -p joepwd -f exportcontainer.csv
```

This example exports all containers whose name starts with the characters `West`.

```
$INCHOME/etc/cli/ExportContainer.sh –u joe –p joepwd –f exportcontainer.csv –n "West*"
```

This example exports all containers that have a UDF named `customer` with the value `12345`.

```
$INCHOME/etc/cli/ExportContainer.sh –u joe –p joepwd –f exportcontainer.csv
–d customer=12345
```

File Format

The format for the export file is as follows.

Note: This is the same format that the **`ImportContainer`** CLI requires.

Col	Field	Accepted Values
A	Container Name	The name of the container.
B	Container Description	A brief description of the container.
C	Parent Container	The name of the parent container for this container.
D	Container Type	Either logical or device.
E	Rule1	A listing of the valid block types for this container, separated by '/'. Block types that with an associated information template have the template name concatenated to the block type name, separated by a '|'.
F	Rule2	A listing of the block types enabled for root block creation, separated by '/'.
G	Rule3	A listing of the block types that can be used for space allocation from the parent container, separated by '/'.
H	Rule4	A listing of the block types for which SWIP Names are required, separated by '/'.
I	Rule5	A listing of the valid device types for this container, separated by '/'. Device types that with an associated information template have the template name concatenated to the device type name, separated by a '|'. If no device types are allowed, the list will contain the keyword **NONE**.
J	Information Template	A listing of information templates associated with this container, separated by '|'.
K	User Defined Fields	User defined field values, in name=value format. Multiple values are separated by a vertical bar ("|"). For example: fieldOne=valueOne|fieldTwo=valueTwo If the UDF type is Checkbox, the valid values are "on" or "off".
L	Maintain History Records	Indicates whether or not Container History and Block History records will be kept for all appropriate block types. The history records are created each time the Global Utilization Rollup task is run. Accepted values are "true" or "false".

ExportDevice

Overview

The **ExportDevice** CLI exports a list of all the Devices into a specified file. This file can be modified and then imported using the **ImportDevice** CLI.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.ExportDeviceCLI -u <userId> -p <pswd>
[-f <export filename>] [-i ipaddress] [-n deviceName] [-d udfName=value] [-m domainName]
[-r ipaddress1/ipaddress2] [-e devicetype] [-b blockName] [-c containerName]
[-t blocktype] [-ex] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ExportDevice.sh -u <userId> -p <pswd>
[-f <export filename>] [-i ipaddress] [-n deviceName] [-d udfName=value] [-m domainName]
[-r ipaddress1/ipaddress2] [-e devicetype] [-b blockName] [-c containerName]
[-t blocktype] [-ex] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ExportDevice.cmd -u <userId> -p <pswd>
[-f <export filename>] [-i ipaddress] [-n deviceName] [-d udfName=value] [-m domainName]
[-r ipaddress1/ipaddress2] [-e devicetype] [-b blockName] [-c containerName]
[-t blocktype] [-ex] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <export filename>	No	The name of the file to export the data to. If no file name is specified, results are outputted to the screen.
-at [addressType]	No	Filter. Allows you to filter export based on address type. Accepted values are: Static, Dynamic DHCP, Automatic DHCP, Manual DHCP, Interface and Reserved. For DHCP V6: Dynamic NA DHCPv6, Automatic NA DHCPv6, Manual NA DHCPv6, Dynamic TA DHCPv6 and Automatic TA DHCPv6.
-b [blockName]	No	Filer. Allows you to filter export based on block name.
-c [containerName]	No	Filter. Allows you to filter export based on the name of the container where the device is located. You may use a fully qualified container name or partial container name with the '*' wildcard. A combination of fully qualified container name and '*' wildcard is not supported.

Parameter	Required	Description
-d [udfName=value]	No	Filter. Allows you to filter export based on the user defined name and value of the device. For wildcarding, use a "*" character within "double quotes", i.e., –d "udfName=value*". Also surround the parameter with double quotes if there is an embedded blank in the filter string. Note that filtering by UDF of value data type URL is not supported.
-de [description]	No	Filter. Allows you to filter export based on the device description.
-e [devicetype]	No	Filter. Allows you to filter export based the user device type name.
-ex	No	Expanded mode. Additional columns will be appended for each UDF defined in the exported records. The first row of the file will be a comment header row, beginning with "^", with each column's filed name, including the tag name for each UDF. The architected UDF column L will contain "Expanded". See below for an example.
-h	No	Filter. *Must be accompanied with –c parameter.* Specifies to recursively include all devices contained in child containers of specified –c parameter filter.
-hw [mac address]	No	Filter. Allows you to filter export based on the MAC Address.
-i [ipaddress]	No	Filter. Allows you to filter export based the ipaddress.
-m [domain]	No	Filter. Allows you to filter export based the domain name.
-n [name]	No	Filter. Allows you to filter export based on the device hostname.
-r [ipaddress1/ipaddress2]	No	Filter. Allows you to filter export based on an address range between ipaddress1 and ipaddress2 inclusive.
-t [blocktype]	No	Filter. Allows you to filter export based on block type name.
-v	No	Produces verbose output.
-vf [true|false]	No	Filter. Allows you to filter export based on virtual flag.
-ch [circuitId]	No	Filter. Allows you to filter export based on the relay agent circuit ID in hex encoded format.
-rh [remoteId]	No	Filter. Allows you to filter export based on the relay agent remote ID in hex encoded format.
-ca [circuitId]	No	Filter. Allows you to filter export based on the relay agent circuit ID in ASCII format.
-ra [remoteId]	No	Filter. Allows you to filter export based on the relay agent remote ID in ASCII format.

Usage Examples

This example exports all devices to file named *exportdevice.csv* in the current directory.

```
$INCHOME/etc/cli/ExportDevice.sh –u joe –p joepwd –f exportdevice.csv
```

This example exports devices that contain an IP address within the range of `10.0.0.1` through `10.0.0.200` inclusive, to the file *exportdevice.csv*.

```
$INCHOME/etc/cli/ExportDevice.sh –u joe –p joepwd –f exportdevice.csv
–r 10.0.0.1/10.0.0.200
```

This example exports devices that are located in the container named `Exton` to the file *exportdevice.csv*.

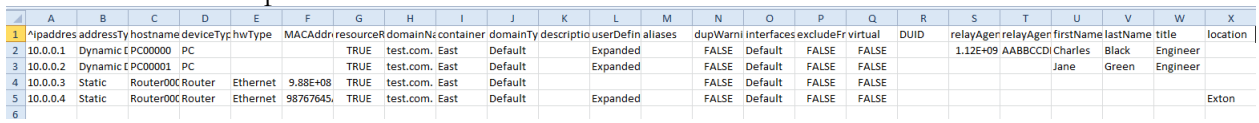
```
$INCHOME/etc/cli/ExportDevice.sh –u joe –p joepwd –f exportdevice.csv –c Exton
```

Export using expanded format example

This example exports devices in the container named Pennsylvania, with the output in expanded format.

```
$INCHOME/etc/cli/ExportDevice.sh -u joe -p joepwd -f cexportdevice.csv -c Pennsylvania
-ex
```

Here is the output file:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1	^ipaddres	addressTy	hostname	deviceTyp	hwType	MACAddr	resourceR	domainNa	container	domainTy	descriptio	userDefin	aliases	dupWarni	interfaces	excludeFr	virtual	DUID	relayAgen	relayAgen	firstName	lastName	title	location
2	10.0.0.1	Dynamic D	PC00000	PC			TRUE	test.com.	East	Default		Expanded		FALSE	Default	FALSE	FALSE		1.12E+09	AABBCCDI	Charles	Black	Engineer	
3	10.0.0.2	Dynamic D	PC00001	PC			TRUE	test.com.	East	Default		Expanded		FALSE	Default	FALSE	FALSE				Jane	Green	Engineer	
4	10.0.0.3	Static	Router00(	Router	Ethernet	9.88E+08	TRUE	test.com.	East	Default				FALSE	Default	FALSE	FALSE							
5	10.0.0.4	Static	Router00(	Router	Ethernet	98767645/	TRUE	test.com.	East	Default		Expanded		FALSE	Default	FALSE	FALSE							Exton
6																								

Note the following:

- The first row is a header line, beginning with "^". The column headers beyond the architected column T contain the user defined field tags.
- Column L contains "Expanded" when the row includes user defined fields beyond column T. Row 4 has no user defined fields defined for that device.
- Only those user defined field tags defined for the devices exported (based on container and deviceType) will have columns in the output file.
- In this example, columns U-X contain the values for the user defined fields. If there is no value, as in row 4 column X, the cell will be blank. In this example, when deviceType=Router, column X is used, as shown in line 5. When deviceType=PC, U-W are used, as shown in lines 2 and 3.
- You can use ImportDevice to import an expanded format exported file by using the –o (overwrite) parameter.
- For devices with multiple IP Addresses on a single interface (devices with address type Interface), use one address to locate the device to export. If this exported output is used for ImportDevice with overwrite, note that device attributes can be modified, but an Interface address cannot be modified or deleted. Use ImportChildBlock with overwrite to modify or delete an Interface address.

File Format

The format for the export file is as follows.

Note: This is the same format that the **`ImportDevice`** CLI requires.

Col	Field	Accepted Values	Required
A	IP Address	The IP address of the device and its device interfaces delimited by '|'.	Yes
B	Address Type	The Address type of the device.	Yes
C	Hostname	The hostname of the device.	Yes
D	Device Type	The device's device type.	Yes
E	Hardware Type	The device's hardware type.	No
F	Mac Address	The Mac Address of the device and its device interfaces delimited by '|'	No
G	Resource Record Flag	"true" if the device is to automatically create an 'A' or 'AAAA' record upon import, 'false' if not.	No
H	Domain Name	The device's domain name.	No
I	Container Name	The container name the device is located in. When there are multiple IP addresses on the device in different containers, multiple containers are delimited by '|'.	Yes
J	Domain Type	The device's domain type.	No
K	Description	The device's description.	No
L	User Defined Fields	The device's user defined field and values delimited by '|'. When this column contains the word "Expanded", the –ex option was invoked. See below for an example of the results.	No
M	Aliases	The device's alias names	No
N	Duplicate warning	'False' if this is a duplicate, 'true' if not.	No
O	Interface Names	The device's interface names	No
P	Exclude from Discovery flags	The flags for excluding the device's IP addresses from host discovery.	No
Q	Virtual	Indicates a virtual address	No
R	DUID	DHCP Unique Identifier	No
S	Relay Agent Circuit Id	Populated after a DHCP Collections task is run.	No
T	Relay Agent Remote Id	Populated after a DHCP Collections task is run.	No

ExportDeviceResourceRecord

Overview

The **ExportDeviceResourceRecord** CLI exports a list of the resource records for a device or group of devices into a specified file. This export will only include resource records (regardless of type) that are visible on the device's Resource Record tab within the user interface. This file can be modified and then used with the **ImportDeviceResourceRecord** CLI or the **ModifyDeviceResourceRecord** CLI.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.ExportDeviceResourceRecordCLI -u <userId> -p <pswd>
[-f <export filename>] [-i ipaddress] [-n deviceName] [-d udfName=value] [-m domainName]
[-a domaintype] [-r ipaddress1/ipaddress2] [-e devicetype] [-b blockName]
[-c containerName] [-t blocktype] [-h] [-o option] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ExportDeviceResourceRecord.sh -u <userId> -p <pswd>
[-f <export filename>] [-i ipaddress] [-n deviceName] [-d udfName=value] [-m domainName]
[-a domaintype] [-r ipaddress1/ipaddress2] [-e devicetype] [-b blockName]
[-c containerName] [-t blocktype] [-h] [-o option] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ExportDeviceResourceRecord.cmd -u <userId> -p <pswd>
[-f <export filename>] [-i ipaddress] [-n deviceName] [-d udfName=value] [-m domainName]
[-a domaintype] [-r ipaddress1/ipaddress2] [-e devicetype] [-b blockName]
[-c containerName] [-t blocktype] [-h] [-o option] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <export filename>	No	The name of the file to export the data to. If no file name is specified, results are outputted to the screen.
-c <containerName>	No	Filter. Allows you to filter export based on the name of the container where the device is located. You may use a fully qualified container name or partial container name with the '*' wildcard. A combination of fully qualified container name and '*' wildcard is not supported.
-d <udfName=value>	No	Filter. Allows you to filter export based the user defined name and value the device owns. Note that filtering by UDF of value data type URL is not supported.
-e <devicetype>	No	Filter. Allows you to filter export based the user device type name.
-i <ipaddress>	No	Filter. Allows you to filter export based the device's IP Address.
-m <domain>	No	Filter. Allows you to filter export based on the domain name of the device.

Parameter	Required	Description
-a <domaintype>	No	Filter. Allows you to filter export based on the domain type of the device.
-n <name>	No	Filter. Allows you to filter export based on the device hostname.
-r <ipaddress1/ipaddress2>	No	Filter. Allows you to filter export based on an address range between ipaddress1 and ipaddress2 inclusive.
-t <blocktype>	No	Filter. Allows you to filter export based on block type name.
-b <blockName>	No	Filter. Allows you to filter export based on block name.
-h	No	Filter. *Must be accompanied with –c parameter.* Specifies to recursively include all devices contained in child containers of specified –c parameter filter.
-o <option>	No	To specify that the export output should be in the format used by ImportDeviceResourceRecord, use **I**. Use **M** for the ModifyDeviceResourceRecord format. The default is **I**.
-s <batch-size>	No	This is an internal buffering parameter to avoid out of memory situations. There can be a very large number of resource records exported with this command. The default batch size for each call to the web service is 1000 devices. If you receive the error, "*Exception in thread "main" java.lang.OutOfMemoryError: Java heap space*", you can use this parameter to specify a smaller batch size, for example, -s 500. You can also edit the command file and change the –Xmx parameter.
-v	No	Produces verbose output.

Usage Examples

This example exports all device resource records (that appear on the device's Resource Record tab within the user interface) for the device at `10.0.0.24`, to a file named *exportdevicerr.csv* in the current directory. The output is formatted for the **`ImportDeviceResourceRecord`** CLI.

```
$INCHOME/etc/cli/ExportDeviceResourceRecord.sh –u joe –p joepwd –i 10.0.0.24
–f exportdevicerr.csv
```

This example exports all device resource records (that appear on the device's Resource Record tab within the user interface) for the device with hostname `switch00024` to a file named *exportdevicerr.csv* in the current directory. The output is formatted for the **`ModifyDeviceResourceRecord`** CLI.

```
$INCHOME/etc/cli/ExportDeviceResourceRecord.sh –u joe –p joepwd –n switch00024
–f exportdevicerr.csv –o M
```

This example exports all device resource records (that appear on a device's Resource Record tab within the user interface) for devices located in container `Exton` to a file named *exportdevicerr.csv* in the current directory. The output is formatted for the **`ModifyDeviceResourceRecord`** CLI.

```
$INCHOME/etc/cli/ExportDeviceResourceRecord.sh –u joe –p joepwd –c Exton exportdevicerr.csv
–o M
```

File Format

The format for the export file when "–o I" is specified is as follows.

Note: This is the same format that the **`ImportDeviceResourceRecord`** CLI requires.

Col	Field	Accepted Values
A	Domain	The name of the domain to which the resource records belongs.
B	Domain Type	The name of the domain type to which the domain belongs. Defaults to "Default"
C	Owner	The OWNER section of the resource record. Note that this section is specific to the type of resource record. Refer to the appropriate RFC for exact text that should be entered.
D	Host Name	The device host name.
E	IP Address	The IP Address of the Device.
F	Container	The name of the container that holds the device. This is required only if there is overlapping address space in use and the ip address is in overlapping space. The container is then used to uniquely determine the device.
G	TTL	The Time To Live.
H	Class	The value currently supported is **IN**. If not specified, defaults to **IN**.
I	Resource Record Type	The type of resource record.
J	Data	The text for the DATA area of the resource record. Note that this section is specific to the type of resource record. Refer to the appropriate RFC for exact text that should be entered.
K	Comment	Text appended to the resource record.

The format for the export file when "–o M" is specified will be of the format that the **`ModifyDeviceResourceRecord`** CLI requires.

For example, suppose a device at 10.0.0.24 has 4 resource records (that appear on the device's Resource Record tab within the user interface). The set of attribute-value pairs before the colon identifies the record to be changed. The set following the colon specifies the values to be changed. All of the values that can be modified are exported.

```
ipAddress=10.0.0.24,container=InControl/Texas/Dallas,owner=switch00026,domain=mycompany.com
,domainType=Default,resourceRecType=A:owner=switch00026,resourceRecType=A,TTL=2400,data=10.
0.0.24,domain=mycompany.com,domainType=Default,comment=

ipAddress=10.0.0.24,container=InControl/Texas/Dallas,owner=24.0.0,domain=10.in-addr.arpa,do
mainType=Default,resourceRecType=PTR:owner=24.0.0,resourceRecType=PTR,TTL=2400,data=switch0
0026.mycompany.com,domain=10.in-addr.arpa,domainType=Default,comment=

ipAddress=10.0.0.24,container=InControl/Texas/Dallas,owner=switch00026,domain=mycompany.com
,domainType=External,resourceRecType=A:owner=switch00026,resourceRecType=A,TTL=2400,data=10
.0.0.24,domain=mycompany.com,domainType=External,comment=

ipAddress=10.0.0.24,container=InControl/Texas/Dallas,owner=24.0.0,domain=10.in-addr.arpa,do
mainType=External,resourceRecType=PTR:owner=24.0.0,resourceRecType=PTR,TTL=2400,data=switch
00026.mycompany.com,domain=10.in-addr.arpa,domainType=External,comment=
```

ExportDeviceRestoreList

Overview

The **ExportDeviceRestoreList** CLI exports a list of the devices that were deleted and are available to be restored. If an admin with superuser role executes the CLI, resource records deleted by any user are available for export. If the admin doesn't have superuser role, then only the records deleted by the admin executing the CLI will be available.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH com.diamondip.ipcontrol.cli.ExportDeviceRestoreListCLI
-u <userId>
-p <pswd> [-f <export filename>] [-ad <Admin>] [-n <hostname>] [-i IPAddress] [-at address
Type] [-r ipaddress1/ipaddress2] [-e devicetype] [-hw HWAddress] [-b blockName] [-c
containerName] [-ex] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ExportDeviceRestoreList.sh -u <userId>
-p <pswd> [-f <export filename>] [-ad <Admin>] [-n <hostname>] [-i IPAddress] [-at address
Type] [-r ipaddress1/ipaddress2] [-e devicetype] [-hw HWAddress] [-b blockName] [-c
containerName] [-ex] [-v] [-?]
```

```
Via command script (Windows)
```

```
%INCHOME%/etc/cli/ExportDeviceRestoreList.cmd -u <userId>
-p <pswd> [-f <export filename>] [-ad <Admin>] [-n <hostname>] [-i IPAddress] [-at address
Type] [-r ipaddress1/ipaddress2] [-e devicetype] [-hw HWAddress] [-b blockName] [-c
containerName] [-ex] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <export filename>	No	The name of the file to export the data to. If no file name is specified, results are outputted to the screen.
-ad <Admin>	No	Filter Allows a **Superuser** to filter export based on the login id of a user that deleted the records. Specify the login id or use the wildcard character '*' to perform a partial match on the login id. Normal users are allowed to only export records that they deleted
-n <name>	No	Filter. Allows you to filter export based on the device hostname. Specify the hostname or use the wildcard character '*' to perform a partial match on the hostname.
-i <ipaddress>	No	Filter. Allows you to filter export based on the device's IP Address.
-a <addresstype>	No	Filter. Allows you to filter export based on the device's address type.
-r <ipaddress1/ipaddress2>	No	Filter. Allows you to filter export based on an address range between ipaddress1 and ipaddress2 inclusive.

Parameter	Required	Description
-e <devicetype>	No	Filter. Allows you to filter export based on the user device type name. Specify the device type or use the wildcard character '*' to perform a partial match on the device type.
-hw <hwaddress>	No	Filter. Allows you to filter export based on the hardware(MAC) address of the device. Specify the hardware address or use the wildcard character '*' to perform a partial match on the hardware address.
-b <blockName>	No	Filter. Allows you to filter export based on block name. Specify the block name or use the wildcard character '*' to perform a partial match on the block name.
-c <containerName>	No	Filter. Allows you to filter export based on the name of the container the device is located in. You may use a fully qualified container name or partial container name with the '*' wildcard. A combination of fully qualified container name and '*' wildcard is not allowed.
-v	No	Produces verbose output.

Usage Example

This example exports all devices with an IP Address 10.0.16.3 that administrator `joe` deleted to a file named *exportrestore.csv* in the current directory.

```
$INCHOME/etc/cli/ExportDeviceRestoreList.sh -u joe -p joepwd -i 10.0.16.3
-f exportrestore.csv

Sample file contents are shown below:

7,10.0.16.3,cmts00003,10.0.16.0/23,CANADA,CMTS,Dynamic
DHCP,,,ethernet,00000000FF00,,,joe,FALSE
```

This example exports all devices in container names starting with 'CAN' that administrator `joe` deleted to a file named *exportrestorecanada.csv* in the current directory.

```
$INCHOME/etc/cli/ExportDeviceRestoreList.sh -u joe -p joepwd -c CAN*
-f exportrestore.csv

Sample file contents are shown below:

7,10.0.16.3,cmts00003,10.0.16.0/23,CANADA,CMTS,Dynamic
DHCP,,,ethernet,00000000FF00,,,joe,FALSE
```

File Format

The format for the export file is described in the table following.

Each line starts with the restore information:

- restore id
- date and time the record was deleted
- login id of the administrator who deleted the device

The remaining columns provide a full description of the deleted device.

ExportDeviceRestoreList

Col	Field	Accepted Values
A	Restore id	The id used internally to identify the record. It is required to restore this record via RestoreDeletedDevice.
B	Date/Time	The date and time when the device was deleted.
C	Administrator	The login id of the administrator who deleted the record.
D	IP Address	The ipaddress of the deleted device.
E	Hostname	The hostname of the deleted device.
F	Block name	The name of the block that the device was deleted from.
G	Container	Name of the Container from which the device was deleted.
H	Device Type	Device type of the deleted device
I	Address Type	Address type of the deleted device.
J	Description	Description of the deleted device.
K	DUID	DHCP Unique Identifier
L	Hardware type	Hardware type of the deleted device. Ethernet or Token Ring
M	Hardware Address	The Hardware(Mac)Address of the deleted device
N	Domain Name	Domain name of the deleted device.
O	Domain Type	Domain type of the deleted device.
P	Ignore Duplicate Warning flag	This field is used by RestoreDeletedDevice CLI. Export puts a false by default for this. Change this to True before using this output for restoring to ignore Duplicate Warnings for mac address or hostname.

ExportDomainResourceRecord

Overview

The **ExportDomainResourceRecord** CLI exports a list of the resource records for a domain or group of domains into a specified file. The default behavior of this CLI is to export only those resource records associated with the specified domains and not bound to a device . This file can then be used with the **ImportDomainResourceRecord** CLI or the **ModifyDomainResourceRecord** CLI. These CLIs allow the administrator to create and modify resource records that are not bound to a particular device.

When the "-id" option (Include Device RRs) is specified , this export will include all of the resource records that are visible on the domain's Resource Record tab in the IPAM user interface, including those bound to a device. Be aware that if you re-import resource records bound to a device using the **ImportDomainResourceRecord** CLI, the association with the device will be lost.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.ExportDomainResourceRecordCLI -u <userId> -p <pswd>
[-f <export filename>] [-d udfName=value] [-m domainName] [-a domaintype]
[-ow owner][-rr type] [-rd data] [-o option] [-id] [-s batch-size] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ExportDomainResourceRecord.sh -u <userId> -p <pswd>
[-f <export filename>] [-d udfName=value] [-m domainName] [-a domaintype]
[-ow owner][-rr type] [-rd data] [-o option] [-id] [-s batch-size] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ExportDomainResourceRecord.cmd -u <userId> -p <pswd>
[-f <export filename>] [-d udfName=value] [-m domainName] [-a domaintype]
[-ow owner][-rr type] [-rd data] [-o option] [-id] [-s batch-size] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <export filename>	No	The name of the file to export the data to. If no file name is specified, results are outputted to the screen.
-d <udfName=value>	No	Filter. Allows you to filter export based the user defined name and value associated with the domain. Specify a User Defined Field (UDF) attached to a domain. The UDF is specified using the syntax "name=value", where name is the field name (tag) of the UDF, and value is the value of the UDF. For wildcarding, use a "*" character within "double quotes", i.e., –d "udfName=value*". Also surround the parameter with double quotes if there is an embedded blank in the filter string. Note that filtering by UDF of value data type URL is not supported.

Parameter	Required	Description
-m <domain name>	No	Filter. Allows you to filter export based on the domain name of the domain that owns the resource record. Specify the name of a specific domain or use the wildcard character '*' to perform a partial match
-a <domain type>	No	Filter. Allows you to filter export based on the domain type of the domain that owns the resource record. Specify the name of a specific domain type or use the wildcard character '*' to perform a partial match
-ow <owner>	No	Filter. Allows you to filter export based on the owner associated with the resource record. Specify the owner or use the wildcard character '*' to perform a partial match.
-rr <type>	No	Filter. Allows you to filter export based on the resource record type associated with the record, for example: -rr A; -rr PTR.
-rd <data>	No	Filter. Allows you to filter export based on the data portion of the resource record. Specify the entire value, or use the wildcard character '*' to perform a partial match.
-id	No	To indicate that all resource records for the included domains should be exported, including those bound to a device. The default behavior is to export only those resource records **not** bound to a device.
-o <option>	No	To specify that the export output should be in the format used by ImportDomainResourceRecord, use **I**. Use **M** for the ModifyDomainResourceRecord format. The default is **I**.
-s <batch-size>	No	This is an internal buffering parameter to avoid out of memory situations. There can be a very large number of resource records exported with this command. The default batch size for each call to the web service is 1000 records. If you receive the error, "*Exception in thread "main" java.lang.OutOfMemoryError: Java heap space*", you can use this parameter to specify a smaller batch size, for example, -s 500. You can also edit the command file and change the –Xmx parameter.
-v	No	Produces verbose output.

Usage Examples

This example exports all domain resource records for the domain `example.com`, to a file named *exportdomainrr.csv* in the current directory. The output is formatted for the **`ImportDomainResourceRecord`** CLI.

```
$INCHOME/etc/cli/ExportDomainResourceRecord.sh –u joe –p joepwd –m example.com
–f exportdomainrr.csv
```

This example exports all domain resource records for domains of type `Internal` to a file named *exportdomainrr.csv* in the current directory. The output is formatted for the **`ModifyDomainResourceRecord`** CLI.

```
$INCHOME/etc/cli/ExportDomainResourceRecord.sh –u joe –p joepwd –a Internal
–f exportdomainrr.csv –o M
```

File Format

- The format for the export file when "–o I" is the same format that the **ImportDomainResourceRecord** CLI requires.
- The format for the export file when "–o M" is specified will be of the format that the **ModifyDomainResourceRecord** CLI requires.

  Below is sample output for the modify format. The set of attribute-value pairs before the "#" identifies the record to be changed. The set following the "#" specifies the values to be changed.

```
domain=mycompany.com., domainType=Default, owner=jh, resourceRecClass=IN,
resourceRecType=A, data=10.0.0.24 # domainType=Default, resourceRecClass=IN, owner=jh,
domain=mycompany.com., data=10.0.0.24, resourceRecType=A, comment=this is a comment,
TTL=1800

domain=mycompany.com., domainType=Internal, owner=st, resourceRecClass=IN,
resourceRecType=A, data=10.0.0.25 # domainType=Internal, resourceRecClass=IN, owner=st,
domain=mycompany.com., data=10.0.0.25, resourceRecType=A, TTL=

domain=10.in-addr.arpa, domainType=Default, owner=24.0.0, resourceRecClass=IN,
resourceRecType=PTR, data=jh.mycompany.com, # domainType=Default, resourceRecClass=IN,
owner=24.0.0, domain=10.in-addr.arpa, data=jh.mycompany.com, resourceRecType=PTR, TTL=2400
```

ExportNetElement

Overview

The **ExportNetElement** CLI exports a list of all the Network Elements into a specified file. This file can be modified and then imported using the **ImportNetElement** CLI.

Note: This CLI has been deprecated as of IPAM 8.1.3. Use the **ExportNetworkElement** CLI instead. ExportNetElement support will be removed in a future release.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.ExportNetElementCLI -u <userId> -p <pswd>
[-f <export filename>] [-n <network element name>] [-i <IP address or host name>]
[-vendor <network element vendor>] [-model <network element model>] [-e <element type>]
[-globalSync <Y|N>] [-agent <agent name>] [-c <container name>] [-s batch-size] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ExportNetElement.sh -u <userId> -p <pswd>
[-f <export filename>] [-n <network element name>] [-i <IP address or host name>]
[-vendor <network element vendor>] [-model <network element model>] [-e <element type>]
[-globalSync <Y|N>] [-agent <agent name>] [-c <container name>] [-s batch-size] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ExportNetElement.cmd -u <userId> -p <pswd>
[-f <export filename>] [-n <network element name>] [-i <IP address or host name>]
[-vendor <network element vendor>] [-model <network element model>] [-e <element type>]
[-globalSync <Y|N>] [-agent <agent name>] [-c <container name>] [-s batch-size] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <export filename>	No	The name of the file to export the data to. If no file name is specified, results are outputted to the screen.
-n <network element name>	No	Filter. Allows you to filter export based on specific fields. See file format for accepted values. If the network element name contains embedded spaces, surround the name with double-quotes, for example, “exton closet “.
-i <IP address or host name>	No	Filter. Allows you to filter export based on specific fields. See file format for accepted values.
-vendor <network element vendor>	No	Filter. Allows you to filter export based on specific fields. See file format for accepted values. If the vendor name contains embedded spaces, surround the name with double-quotes.
-model <network element model>	No	Filter. Allows you to filter export based on specific fields. See file format for accepted values. If the model name contains embedded spaces, surround the name with double-quotes.
-e <element type>	No	Filter. Allows you to filter export based on specific fields. See file format for accepted values.

Parameter	Required	Description
-globalSync <Y|N>	No	Filter. Allows you to filter export based on specific fields. See file format for accepted values.
-agent <agent name>	No	Filter. Allows you to filter export based on specific fields. See file format for accepted values. If the agent name contains embedded spaces, surround the name with double-quotes.
-c <container name>	No	Filter. Allows you to filter export based on container. If the container name contains embedded spaces, surround the name with double-quotes, for example, "New York ".
-s	No	The batch size option is not implemented in this CLI.
-v	No	Provides verbose output.

Usage Example

This example exports all network elements to an *neexport.csv* file in the current directory.

```
$INCHOME/etc/cli/ExportNetElement.sh –u joe –p joepwd –f neexport.csv
```

This example exports `Cisco` network elements that are not included in the `GlobalSync` process to the file *neexport.csv*.

```
$INCHOME/etc/cli/ExportNetElement.sh –u joe –p joepwd –f neexport.csv –globalSync N
–vendor "Cisco Systems"
```

File Format

The format for the export file is as follows.

Note: This is the same format that the **`ImportNetElement`** CLI requires.

Col	Field	Accepted Values	Required
A	Name	The name of the Network Element. This can be any combination of letters and numbers.	Yes
B	IP Address/FQDN	The IP address or fully-qualified domain name (FQDN) of the Network Element. This must be a valid IPv4 or IPv6 IP address, or a fully-qualified host name.	Yes
C	Vendor	The vendor of the Network Element.	Yes
D	Model	The model name of the Network Element.	Yes
E	Type	The type of Network Element. Accepted values are **CMTS**, **router**, **switch**, or **vpn**.	Yes
F	Global Sync	Whether or not to include this Network Element in the Global Sync process. Accepted values are **True** or **False** (case insensitive).	Yes
G	Agent Name	The exact name of the IPAM Agent that's responsible for contacting this Network Service.	Yes
H	Telnet User	A user name used to telnet to this device.	No
I	Telnet Password	A password used by the telnet user to telnet to this device.	No
J	Enable Password	Password used to enter "enabled" or "privileged" mode on the device.	No
K	Read Community String	The community string used by SNMP to read details from this network element.	No
L	Interfaces	A list of enabled interfaces.	No

Col	Field	Accepted Values	Required
M	V3 Username	Required if using SNMP V3. Blank if V1 or no SNMP.	No
N	V3 Authentication Protocol	Either **MD5** or **SHA1**. Blank if V1 or no SNMP	No
O	V3 Authentication Password	Authentication password. Blank if V1 or no SNMP	No
P	V3 Privacy Protocol	Privacy Protocol. Blank if V1 or no SNMP	No
Q	V3 Privacy Password	Privacy Password. Blank if V1 or no SNMP	No
R	V3 Context Name	SNMP V3 Context name. Blank if V1 or no SNMP	No
S	V3 Engine ID	SNMP V3 Engine ID. Blank if V1 or no SNMP	No

ExportNetworkElement

Overview

The **ExportNetworkElement** CLI exports a list of all the Network Elements into a specified file. It deprecates ExportNetElement CLI by by adding support for multiple agents and several other fields not previously exported. Telnet Username, Telnet Password, and Enable Password are not implemented as these fields are now deprecated.

This file can be modified and then imported using the **ImportNetworkElement** CLI.

Usage

Direct

```
$INCHOME/jre/bin/java –cp $CLASSPATH
com.diamondip.netcontrol.cli.ExportNetworkElementCLI –u <userId> -p <pswd>
[-f <export filename>] [-n <network element name>] [-i <IP address or host name>]
[-vendor <network element vendor>] [-model <network element model>] [-e <element type>]
[-gs <true|false>] [-agent <agent name 1|agent name 2|…>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ExportNetworkElement.sh –u <userId> -p <pswd>
[-f <export filename>] [-n <network element name>] [-i <IP address or host name>]
[-vendor <network element vendor>] [-model <network element model>] [-e <element type>]
[-gs <true|false>] [-agent <agent name 1|agent name 2|…>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ExportNetworkElement.cmd –u <userId> -p <pswd>
[-f <export filename>] [-n < network element name>] [-i <IP address or host name>]
[-vendor <network element vendor>] [-model <network element model>] [-e <element type>]
[-gs <true|false>] [-agent <agent name>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <export filename>	No	The name of the file to export the data to. If no file name is specified, results are outputted to the screen.
-n <network element name>	No	Filter. Allows you to filter export based on the name of the network element. If the network element name contains embedded spaces, surround the name with double-quotes, for example, “exton closet “. For wildcarding, use a “*” character within “double quotes”, i.e., –n “value*”.
-i <IP address>	No	Filter. Allows you to filter export based on the IP address of the network element. “. For wildcarding, use a “*” character within “double quotes”, i.e., –i “10.0.0.*”.
-ve <network element vendor>	No	Filter. Allows you to filter export based on the vendor of the network element. If the vendor name contains embedded spaces, surround the name with double-quotes. “. For wildcarding, use a “*” character within “double quotes”, i.e., –ve “value*”.

Parameter	Required	Description
-mo <network element model>	No	Filter. Allows you to filter export based on the model of the network element. If the model name contains embedded spaces, surround the name with double-quotes. ". For wildcarding, use a "*" character within "double quotes", i.e., –mo "value*".
-e <element type>	No	Filter. Allows you to filter export based on the network element type.
-gs <true|false>	No	Filter. Allows you to filter export based on whether or not the network element is set to be included in a global synchronization task.
-ag <agent name>	No	Filter. Allows you to filter export by the collection agent assigned to a network element. If the agent name contains embedded spaces, surround the name with double-quotes, for example: "agent name 1". ". For wildcarding, use a "*" character within "double quotes", i.e., –ag "value*".
-s	No	The batch size option is not implemented in this CLI.
-v	No	Provides verbose output.

Usage Example

This example exports all network elements to an *neexport.csv* file in the current directory.

```
$INCHOME/etc/cli/ExportNetworkElement.sh –u joe –p joepwd –f neexport.csv
```

This example exports `Cisco` network elements that are not included in the `GlobalSync` process to the file *neexport.csv*.

```
$INCHOME/etc/cli/ExportNetworkElement.sh –u joe –p joepwd –f neexport.csv –gs false
–ve "Cisco Systems"
```

File Format

The format for the export file is as follows.

Note: This is the same format that the **`ImportNetWorkElement`** CLI requires.

Col	Field	Accepted Values
A	Name	The name of the Network Element. This can be any combination of letters and numbers.
B	Description	A description of the Network Element. "\r\n" is used to separate lines.
C	IP Address	The IP address of the Network Element.
D	Type	The type of Network Element. Values are typically **cmts**, **router**, **switch**, or **vpn,** but any valid device type configured in IPAM may have been specified.
E	Vendor	The vendor of the Network Element.
F	Model	The model name of the Network Element.
G	Agent Name(s)	The name(s) of the IPAM Agent(s) that are responsible for contacting this Network Element. If there multiple agents, their names are separated by vertical bars("|").
H	Global Sync	Whether or not to include this Network Element in the Global Sync process. Value is exported as **true** or **false**.
I	SNMP Version	**V1, V2** or **V3**.

Col	Field	Accepted Values
J	Read Community String	The community string used by SNMP V1, SNMP V2 to read details from this network element.
K	SNMP Timeout	The number of milliseconds the SNMP Agent will wait without receiving any messages from its partner before it assumes that the connection to its partner has failed.
L	SNMP Retries	The number of connection retries that the SNMP Agent will attempt.
M	V3 Username	User name for SNMP V3.
N	V3 Authentication Protocol	**MD5** or **SHA1**
O	V3 Authentication Password	Required if field N is set to either **MD5** or **SHA1**. Always exported as "********".
P	V3 Privacy Protocol	Only **DES** supported at this time.
Q	V3 Privacy Password	Required for privacy protocol **DES**. Always exported as "********".
R	V3 Context Name	SNMP V3 context name.
S	V3 Engine ID	SNMP V3 engine ID.
T	Interface List	The list of interface name/status pair(s) separated by vertical bars ("|"). Status is **enabled**, **disabled** or **deploying**. For example: Ethernet0/enabled | Ethernet1/disabled
U	Device Interface Template and interface status	Applies to Import only.
V	SNMP sysName	Populated after a Discover Router Subnets task is run against the network element. This is a standard MIB-II variables (see RFC-1213).
W	SNMP sysDescr	Populated after a Discover Router Subnets task is run against the network element. This is a standard MIB-II variables (see RFC-1213).
X	SNMP sysLocation	Populated after a Discover Router Subnets task is run against the network element. This is a standard MIB-II variables (see RFC-1213).
Y	SNMP sysServices	Populated after a Discover Router Subnets task is run against the network element. This is a standard MIB-II variables (see RFC-1213).

ExportNetService

Overview

The **ExportNetService** CLI exports a list of all the DHCP Network Services into a specified file. This file can be modified and then imported using the **ImportNetService** CLI.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.ExportNetServiceCLI -u <userId> -p <pswd>
[-f <export filename>] [-name <NetService Name>] [-address <IP Address or host name>]
[-product <Product name>] [-agent <Agent name>]
[-globalsync <Y|N>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ExportNetService.sh -u <userId> -p <pswd>
[-f <export filename>] [-name <NetService Name>] [-address <IP Address or host name>]
[-product <Product name>] [-agent <Agent name>]
[-globalsync <Y|N>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ExportNetService.cmd -u <userId> -p <pswd>
[-f <export filename>] [-name <NetService Name>] [-address <IP Address or host name>]
[-product <Product name>] [-agent <Agent name>]
[-globalsync <Y|N>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <export filename>	No	The name of the file to export the data to, in CSV format. If no file name is specified, results are outputted to the screen.
-name <NetService Name>	No	Filter. Allows you to filter export based on specific fields. See file format for accepted values.
-address <IP Address or host name>	No	Filter. Allows you to filter export based on specific fields. See file format for accepted values.
-product <Product name>	No	Filter. Allows you to filter export based on specific fields. See file format for accepted values. If the product name contains embedded spaces, surround the name with double-quotes, for example, "ISC DHCP 3.x".
-agent <Agent name>	No	Filter. Allows you to filter export based on specific fields. See file format for accepted values. If the agent name contains embedded spaces, surround the name with double-quotes, for example, "Executive Agent".

-c <Container Name>	No	Filter. Allows you to filter export based on container. If the container name contains embedded spaces, surround the name with double-quotes, for example, "New York ".
- globalsync <Y|N>	No	Filter. Allows you to filter export based on specific fields. See file format for accepted values.
-s	No	The batch size option is not implemented in this CLI.
-type	No	The type option is ignored in this CLI.
-v	No	Provides verbose output.

Usage Example

This example exports network services to an *nsexport.csv* file in the current directory.

```
$INCHOME/etc/cli/ExportNetService.sh –u joe –p joepwd –f nsexport.csv
```

This example exports network services using the product `INS DHCP` to the standard output (usually screen).

```
$INCHOME/etc/cli/ExportNetService.sh –u joe –p joepwd –product "INS DHCP"
```

File Format

The format for the export file is as follows.

Note: This is the same format that the **`ImportNetService`** CLI requires.

Col	Field	Accepted Values	Required
A	Name	The name of the Network Service. This can be any combination of letters and numbers.	Yes
B	IP Address/FQDN	The IP address or fully-qualified domain name (FQDN) of the Network Service. This must be a valid IPv4 or IPv6 IP address, or a fully-qualified host name.	Yes
C	Type	The type of Network Service. This is always **dhcp.**	No
D	Product name	The Network Service product name. This must be a value already defined in IPAM, for example**, INS DHCP** or **CNR DHCP**.	Yes
E	Agent name	The exact name of the IPAM Agent that's responsible for contacting this Network Service.	Yes
F	Global Sync	Whether or not to include this Network Service in the Global Sync process. Accepted values are **True** or **False** (case insensitive).	Yes
G	Collection Method	The method by which the IPAM Agent will collect data from the Network Service. Accepted values are **scp** or **ftp**.	Yes
H	User name for collection	The username used by the collection method (scp or ftp) to log in to the remote server. Exported as "username".	Yes
I	Password for collection	The password used by the collection method (scp or ftp) to log in to the remote server. Used in conjunction with the 'User name for collection'. Exported as "password".	Yes

Col	Field	Accepted Values	Required
J	Collection port	The port number the collection method (scp or ftp) is listening on. If no value is specified, this will default to **22** if the collection method is scp, and **21** if the collection method is ftp.	No
K	Container(s)	No longer used.	
L	VendorInfo	Vendor specific information for the product's collection type. Each item of information is specified in this single field by separating each field with the '|' character. For collection types **qip**, **adc**, **msft** and **isc**, the information includes the DHCP Configuration file pathname and DHCP Active Lease file pathname. For example, /opt/qip/dhcp/dhcpd.conf |/opt/qip/dhcp/dhcp.db or c:\qip\dhcp\dhcpd.conf |c:\qip\dhcp\dhcp.db For collection type **cnr**, the information includes the Path/Executable of NRCD command, the NRCMD user id, the NRCMD password and the Cluster Name. For example, /opt/cnr/bin/nrcmd|myuserid|mypass| cluster1	No
M	WarningThreshold	Default scope utilization warning threshold. Provide warnings when usage of a pool assigned to this service is exceeded. If no value is specified, this will default to **90.**	No

Note: The CLI does **not** export columns H or I since these could contain sensitive information. The columns are preserved to maintain conformity with the **`ImportNetService`** CLI.

ExportNetworkLink

Overview

The **ExportNetworkLink** CLI exports a list of Network Links into a specified file. This file can be modified and then imported using the **ImportNetworkLink** CLI.

Usage

Direct

```
$INCHOME/jre/bin/java –cp $CLASSPATH
com.diamondip.ipcontrol.cli.ExportNetworkLinkCLI –u <userId>  -p <pswd>
[-ln <link name>] [-lt <link type>] [-de <description>] [-n <block name>]
[-c container][-f <export-file>] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ExportNetworkLink.sh –u <userId>  -p <pswd>
[-ln <link name>] [-lt <link type>] [-de <description>] [-n <block name>]
[-c container] [-f <export-file>] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ExportNetworkLink.cmd –u <userId>  -p <pswd>
[-ln <link name>] [-lt <link type>] [-de <description>] [-n <block name>]
[-c container] [-f <export-file>] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <export filename>	No	The name of the file to export the data to. If no file name is specified, results are outputted to the screen.
-ln [link name]	No	Filter. Allows you to filter export based on the network link name. Surround the parameter with double quotes if there is an embedded blank, for example, -ln “two words”.
-lt [link type]	No	Filter. Allows you to filter export based on the network link type. Specify **logical** or **physical**.
-de [description]	No	Filter. Allows you to filter export based on the network link description. Surround the parameter with double quotes if there is an embedded blank, for example, -de “two words”.
-n <block name>	No	Filter. Specify the name of a specific block or use the wildcard character ‘*’ to perform a partial match on the name. If the block name contains embedded spaces, surround the name with double-quotes “ “.
-c <container name>	No	Filter. Specify the name of a specific container to disambiguate the block in overlapping space specified in the –n parameter. You may use a fully qualified container name or partial container name with the ‘*’ wildcard. A combination of fully qualified container name and ‘*’ wildcard is not supported. This parameter is only valid when the block name is also specified.

Usage Example

This example exports all network links to the file *nlexport.csv* in the current directory.

```
$INCHOME/etc/cli/ExportNetworkLink.sh -u joe -p joepwd -f nlexport.csv
```

This example exports network links that begin with the name `MyLink` and that are of link type `Logical` to the file *nlexport.csv*.

```
$INCHOME/etc/cli/ExportNetworkLink.sh -u joe -p joepwd -f nlexport.csv -ln MyLink*
-lt Logical
```

File Format

This CLI uses the same format that the **`ImportNetworkLink`** CLI requires.

ExportPrefixPool

Overview

The **ExportPrefixPool** CLI exports a list of all the V6 Prefix Pools into a specified file. This file can be modified and then imported using the **ImportPrefixPool** CLI.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.ExportPrefixPoolCLI -u <userId> -p <pswd>
[-f <export filename>] [-name <Prefix Pool Name>] [-ipaddress <IP Address>]
[-container <Container Name>] [-ipaddressrange <Contained IP Address Range
IPAddress/length>] [-netservice <DHCP Server Name>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ExportPrefixPool.sh -u <userId> -p <pswd>
[-f <export filename>] [-name <Prefix Pool Name>] [-ipaddress <IP Address>]
[-container <Container Name>] [-ipaddressrange <Contained IP Address Range
IPAddress/length>] [-netservice <DHCP Server Name>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ExportPrefixPool.cmd -u <userId> -p <pswd>
[-f <export filename>] [-name <Prefix Pool Name>] [-ipaddress <IP Address>]
[-container <Container Name>] [-ipaddressrange <Contained IP Address Range
IPAddress/length>] [-netservice <DHCP Server Name>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	No	The name of the file to export the data to, in CSV format. If no file name is specified, results are outputted to the screen.
-name <Prefix Pool Name>	No	Filter. Allows you to filter export based prefix pool name. Specify the prefix pool name or use the wildcard character '*' to perform a partial match on the prefix pool name.
-ipaddress <IP Address>	No	Filter. Allows you to filter export based on what pools contain this specific address.
-container <Container name>	No	Filter. Allows you to filter export based on which pools live in a specific container. You may use a fully qualified container name or partial container name with the '*' wildcard. A combination of fully qualified container name and '*' wildcard is not supported.
-ipaddressrange < IP Address /length>	No	Filter. Allows you to filter export based on whether a range of values falls within the prefix pool. The range should be of the format IPAddress/length.

-netservice <DHCP Server Name>	No	Filter. Allows you to filter export based on a DHCP server assigned to that prefix pool. Specify the server name or use the wildcard character '*' to perform a partial match on the server name.

Usage Example

This example exports network services to an *prefixpoolexport.csv* file in the current directory.

```
$INCHOME/etc/cli/ExportPrefixPool.sh –u joe –p joepwd –f prefixpoolexport.csv
```

This example exports prefixpools using the netservice `INS DHCP` to the standard output (usually screen).

```
$INCHOME/etc/cli/ExportNetService.sh –u joe –p joepwd –netservice "INS DHCP"
```

File Format

The format for the export file is as follows.

Note: This is the same format that the **`ImportPrefixPool`** CLI requires.

Col	Field	Accepted Values	Required
A	Start Address	The IP Address of the first address in the pool. This address must be in a block with an In-Use/Deployed status.	Yes
B	Length	The length of the prefix pool.	Yes
C	Address Pool Type	One of "Dynamic PD DHCPv6" or "Automatic PD DHCPv6".	Yes
D	Name	Prefix Pool name. Defaults to "Start Address/Length"	No
E	Container	The name of the container that holds the block in which the pool is defined. This is required only if there is overlapping address space in use, and the start address is in overlapping space. The container is then used to uniquely determine the block that will contain the address pool.	No, unless Start address is not unique.
F	Delegated Prefix Length	The default length of the delegated prefix.	Yes
G	Longest Prefix Length	The longest prefix length allowed for delegated prefixes. CNR DHCP only.	No
H	Shortest Prefix Length	The shortest prefix length allowed for delegated prefixes. CNR DHCP only.	No
I	Primary Net Service	The name of the DHCP server that will serve addresses from this pool	No
J	DHCP Option Set	The name of an Option Set used with this pool.	No
K	DHCP Policy Set	The name of a Policy Set used with this pool.	No
L	Allow DHCP Client Classes	A list of Client Classes that are allowed in this address pool. Separate the list entries with a vertical bar "|". For example, to allow two client classes named "allowA" and "allowB", specify: allowA|allowB	No

Col	Field	Accepted Values	Required
M	Deny DHCP Client Classes	A list of Client Classes that are NOT allowed in this address pools. Separate the list entries with a vertical bar "|". For example, to disallow two client classes named "denyA" and "denyB", specify: denyA|denyB	No
N	OverlapInterfaceIp	Flag to allow a DHCPv6 prefix pool to overlap an interface address. This flag may be set only if pool is managed by a CNR DHCPv6 server.	No

ExportResourceRecordPendingApproval

Overview

The **ExportResourceRecordPendingApproval** CLI exports a list of the resource record updates that are waiting for approval by the invoking administrator. These updates include those to create, update, or delete a resource record. The list of workflow ids included in the exported information can then be used with the **ModifyPendingApproval** CLI by an administrator with approval authority to approve or reject the requested changes.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.ExportResourceRecordPendingApprovalCLI -u <userId> -p <pswd>
[-f <export filename>] [-m <domain>] [-a <domaintype>] [-q <requesting admin>]
[-x <action>] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ExportResourceRecordPendingApproval.sh -u <userId> -p <pswd>
[-f <export filename>] [-m <domain>] [-a <domaintype>] [-q <requesting admin>]
[-x <action>] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ExportResourceRecordPendingApproval.cmd -u <userId> -p <pswd>
[-f <export filename>] [-m <domain>] [-a <domaintype>] [-q <requesting admin>]
[-x <action>] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <export filename>	No	The name of the file to export the data to. If not specified, results are written to the screen.
-m <domain>	No	Filter. Allows you to filter export based on the domain name of the resource record. The domain names of the exported records will contain the specified value.
-a <domaintype>	No	Filter. Allows you to filter export based on the domain type of the resource record.
-q <requesting admin>	No	Filter. Allows an approving administrator to filter the export based on the login id of the administrator requesting the resource record change.
-x <action>	No	Filter. Allows you to filter export based on the request to "create", "update" or "delete" a resource record.
-v	No	Produces verbose output.

Usage Examples

This example exports all resource records that are in a pending approval state for administrator `joe` to a file named *exportpending.csv* in the current directory.

```
$INCHOME/etc/cli/ExportResourceRecordPendingApproval.sh -u joe -p joepwd
-f exportpending.csv
```

Sample file contents are shown below:

```
1018, 2009-07-16 23:45:12,jane,Create,example.com.,Default,
Switch3,1800,IN,CNAME,switch00033.example.com.
1019,2009-07-16 21:23:02,jane,Update,example.com.->new.example.com.,
Default,switch00024,1800->2400,IN,A,10.0.0.3,oldcomment->newcomment
1020, 2009-07-17 21:32:22,tom,Delete,example.com.,Default,
RouterSevenWest,1300,IN,CNAME,router00007.test.com.
```

The following example exports all resource records that are in a pending state for administrator joe, requested by `jane`:

```
$INCHOME/etc/cli/ExportResourceRecordPendingApproval.sh -u joe -p joepwd
-f exportpending.csv -q jane
```

This produces the first two records of the file contents shown in the previous example.

File Format

The format for the export file is described in the table following.

Each line starts with the workflow information:

- workflow id
- date and time the action was requested
- requestor's IPAM administrator's login id
- action requested.

The next two columns contain the domain name and type.

The remaining columns provide a full description of the resource record. For update requests, old and new values are shown as "*old value -> new value*".

Col	Field	Accepted Values
A	Workflow id	The id required to approve/reject this change via ModifyPendingApproval.
B	Date/Time	The date and time of the approval request.
C	Administrator	The login id of the administrator submitting the request for approval.
D	Action	One of "Create, Update, Delete"
E	Domain	The name of the domain to which the resource records belongs.
F	Domain Type	The name of the domain type to which the domain belongs.
G	Owner	The OWNER section of the resource record. Note that this section is specific to the type of resource record. Refer to the appropriate RFC for exact text format.
H	TTL	The Time To Live.
I	Class	The value currently supported is **IN**.

J	Resource Record Type	The type of resource record.
K	Data	The text for the DATA area of the resource record. Note that this section is specific to the type of resource record. Refer to the appropriate RFC for exact text format.
L	Comment	Text appended to the resource record.

ExportResourceRecordPendingApprovalStatus

Overview

The **`ExportResourceRecordPendingApprovalStatus`** CLI exports a list of the resource record updates that were submitted for approval by the invoking administrator. These updates include those to create, update, or delete a resource record.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.ExportResourceRecordPendingApprovalStatusCLI -u <userId>
-p <pswd> [-f <export filename>] [-m <domain>] [-a <domaintype>] [-x <action>]
[-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ExportResourceRecordPendingApprovalStatus.sh -u <userId>
-p <pswd> [-f <export filename>] [-m <domain>] [-a <domaintype>] [-x <action>]
[-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ExportResourceRecordPendingApprovalStatus.cmd -u <userId>
-p <pswd> [-f <export filename>] [-m <domain>] [-a <domaintype>] [-x <action>]
[-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <export filename>	No	The name of the file to export the data to. If no file name is specified, results are written to the screen.
-m <domain>	No	Filter. Allows you to filter export based on the domain name of the resource record. The domain names of the exported records will contain the specified value.
-a <domaintype>	No	Filter. Allows you to filter export based on the domain type of the resource record.
-x <action>	No	Filter. Allows you to filter export based on the request to "create", "update" or "delete" a resource record.
-v	No	Produces verbose output.

Usage Example

This example exports all resource records that administrator `joe` submitted for approval to a file named *exportstatus.csv* in the current directory.

```
$INCHOME/etc/cli/ExportResourceRecordPendingApprovalStatus.sh -u joe -p joepwd
-f exportstatus.csv
```

Sample file contents are shown below:

```
1018, 2009-07-16 23:45:12,joe,Create,example.com.,Default,
Switch3,1800,IN,CNAME,switch00033.example.com.
1019,2009-07-16 21:23:02,joe,Update,example.com.->new.example.com.,
Default,switch00024,1800->2400,IN,A,10.0.0.3,oldcomment->newcomment
1020, 2009-07-17 21:32:22,joe,Delete,example.com.,Default,
RouterSevenWest,1300,IN,CNAME,router00007.test.com.
```

File Format

The format for the export file is described in the table following.

Each line starts with the workflow information:

- workflow id
- date and time the action was requested
- requestor's IPAM administrator's login id (same as invoking administrator)
- action requested.

The next two columns contain the domain name and type.

The remaining columns provide a full description of the resource record. For update requests, old and new values are shown as "*old value -> new value*".

Col	Field	Accepted Values
A	Workflow id	The id required to approve/reject this change via ModifyPendingApproval.
B	Date/Time	The date and time of the approval request.
C	Administrator	The login id of the administrator submitting the request (same as invoking administrator).
D	Action	One of "Create, Update, Delete"
E	Domain	The name of the domain to which the resource records belongs.
F	Domain Type	The name of the domain type to which the domain belongs.
G	Owner	The OWNER section of the resource record. Note that this section is specific to the type of resource record. Refer to the appropriate RFC for exact text format.
H	TTL	The Time To Live.
I	Class	The value currently supported is **IN**.
J	Resource Record Type	The type of resource record.
K	Data	The text for the DATA area of the resource record. Note that this section is specific to the type of resource record. Refer to the appropriate RFC for exact text format.
L	Comment	Text appended to the resource record.

ExportResourceRecordRestoreList

Overview

The **`ExportResourceRecordRestoreList`** CLI exports a list of the resource records that were deleted and are available to be restored. If an admin with superuser role executes the CLI, resource records deleted by any user are available for export. If the admin doesn't have superuser role, then only the records deleted by the admin executing the CLI will be available.

Usage

Direct

```
$INCHOME/jre/bin/java –cp $CLASSPATH
com.diamondip.ipcontrol.cli.ExportResourceRecordRestoreListCLI –u <userId>
-p <pswd> [-f <export filename>] [-ad <admin>] [-m <domain>] [-a <domaintype>] [-n
<hostname>] [-i <ipaddress>] [-rd <rdata>] [-rr <rrtype>] [-ow <owner>] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ExportResourceRecordRestoreList.sh –u <userId>
-p <pswd> [-f <export filename>] [-ad <admin>] [-m <domain>] [-a <domaintype>] [-n
<hostname>] [-i <ipaddress>] [-rd <rdata>] [-rr <rrtype>] [-ow <owner>] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ExportResourceRecordRestoreList.cmd –u <userId>
-p <pswd> [-f <export filename>] [-ad <admin>] [-m <domain>] [-a <domaintype>] [-n
<hostname>] [-i <ipaddress>] [-rd <rdata>] [-rr <rrtype>] [-ow <owner>] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <export filename>	No	The name of the file to export the data to. If no file name is specified, results are written to the screen.
-ad <admin>	No	Filter. Allows a superuser to filter export based on the login id of the admin who originally deleted the resource record. Specify the login id or use the wildcard character '*' to perform a partial match on the login id.If a non-superuser uses it an error is generated.
-m <domain>	No	Filter. Allows you to filter export based on the domain name of the deleted resource record. Specify the domain name or use the wildcard character '*' to perform a partial match on the domain name.
-a <domaintype>	No	Filter. Allows you to filter export based on the domain type of the deleted resource record. Specify the domain type or use the wildcard character '*' to perform a partial match on the domain type.
-n <hostname>	No	Filter. Allows you to filter export based on the hostname of the device that the resource record was deleted from. Specify the hostname or use the wildcard character '*' to perform a partial match on the hostname.

-i <ipaddress>	No	Filter. Allows you to filter export based on the ipaddress of the device that the resource record was deleted from.
-rd <rdata>	No	Filter. Allows you to filter export based on the RDATA field of the deleted resource record. Specify the rdate or use the wildcard character '*' to perform a partial match on the rdata.
-rr <rrtype>	No	Filter. Allows you to filter export based on the type field of the deleted resource record. Specify the resource record type name or use the wildcard character '*' to perform a partial match on the resource record type.
-ow <owner>	No	Filter. Allows you to filter export based on the owner field of the deleted resource record. Specify the owner or use the wildcard character '*' to perform a partial match on the owner.
-v	No	Produces verbose output.

Usage Example

This example exports all resource records that administrator `joe` deleted to a file named *exportrestore.csv* in the current directory.

```
$INCHOME/etc/cli/ExportResourceRecordRestoreList.sh –u joe –p joepwd
–f exportrestore.csv
```

```
Sample file contents are shown below:
1018, 2009-07-16 23:45:12,joe,example.com.,Default,
Switch3,1800,IN,CNAME,switch00033.example.com.,,,false
```

File Format

The format for the export file is described in the table following.

Each line starts with the restore information:

- restore id
- date and time the record was deleted
- login id of the administrator who deleted the record

The next two columns contain the domain name and type.

The remaining columns provide a full description of the resource record.

Col	Field	Accepted Values
A	Restore id	The id used internally to identify the record. It is required to restore this record via RestoreDeletedResourceRecord.
B	Date/Time	The date and time when the record was deleted.
C	Administrator	The login id of the administrator who deleted the record.
D	Domain	The name of the domain to which the resource records belongs.
E	Domain Type	The name of the domain type to which the domain belongs.
F	Owner	The OWNER section of the resource record. Note that this section is specific to the type of resource record. Refer to the appropriate RFC for exact text format.
G	TTL	The Time To Live.
H	Class	The value currently supported is **IN**.

I	Resource Record Type	The type of resource record.
J	Data	The text for the DATA area of the resource record. Note that this section is specific to the type of resource record. Refer to the appropriate RFC for exact text format.
K	Comment	Text appended to the resource record.
L	Hostname	Hostname of the device from which the resource record was deleted. Filled only if a device resource record was deleted.
M	IP Address	IP Address of the device from which the resource record was deleted. Filled only if a device resource record was deleted.
N	Ignore Duplicate Warning flag	This field is used by RestoreDeletedResourceRecord CLI. Export puts a false by default for this. Change this to True before using this output for restoring to ignore Duplicate Warnings for owner field.

ExportRootBlock

Overview

The **ExportRootBlock** CLI exports a list of all the Root Blocks into a specified file. This file can be modified and then imported using the **ImportRootBlock** CLI.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.ExportRootBlockCLI -u <userId> -p <pswd>
[-f <export filename>] [-n <block name>] [-b <block address/size>] [-t <block type>]
[-c <container name>] [-i <IP address>] [-r <IP address range>]
[-d <user defined field name=value>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ExportRootBlock.sh -u <userId> -p <pswd>
[-f <export filename>] [-n <block name>] [-b <block address/size>] [-t <block type>]
[-c <container name>] [-i <IP address>] [-r <IP address range>]
[-d <user defined field name=value>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ExportRootBlock.cmd -u <userId> -p <pswd>
[-f <export filename>] [-n <block name>] [-b <block address/size>] [-t <block type>]
[-c <container name>] [-i <IP address>] [-r <IP address range>]
[-d <user defined field name=value>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <export filename>	No	The name of the file to export the data to. If no file name is specified, results are outputted to the screen.
-n <block name>	No	Filter. Specify the name of a specific block or use the wildcard character '*' to perform a partial match on the name. If the block name contains embedded spaces, surround the name with double-quotes " ".
-b <block address/size>	No	Filter. Specify the CIDR notation of a block. That is, specify the starting IP address for the block, followed by the block size, separated by a slash '/'. For example, 10.0.0.0/24. The wildcard character '*' can be used in the start address only.
-t <block type>	No	Filter. Specify the name of a specific block type or use the wildcard character '*' to perform a partial match on the block type name.
-c <container name>	No	Filter. Allows you to filter export based on the name of the container where the block is located. You may use a fully qualified container name or partial container name with the '*' wildcard. A combination of fully qualified container name and '*' wildcard is not supported.
-i <IP address>	No	Filter. Specify an IP address which falls within the start and end address of a block.

Parameter	Required	Description
-r <IP address range>	No	Filter. Specify a range of IP addresses which span one or more blocks. The format of the range is specified as start-end. For example, 10.0.0.0-10.0.255.255.
-d <user defined field name=value>	No	Filter. Specify a User Defined Field (UDF) attached to a block. The UDF is specified using the syntax name=value, where name is the name of the UDF, and value is the value of the UDF. For wildcarding, use a "*" character within "double quotes", i.e., –d "udfName=value*". Also surround the parameter with double quotes if there is an embedded blank in the filter string. Note that filtering by UDF of value data type URL is not supported.

Usage Example

This example exports all root blocks to a *rbexport.csv* file in the current directory.

```
$INCHOME/etc/cli/ExportRootBlock.sh –u joe –p joepwd –f rbexport.csv
```

This example exports root blocks that begin with the name `MyBlock` and that are of block type `Private` to the file *rbexport.csv*.

```
$INCHOME/etc/cli/ExportRootBlock.sh –u joe –p joepwd –f rbexport.csv –n MyBlock*
–t Private
```

File Format

The format for the export file is as follows.

Note: This is the same format that the **`ImportRootBlock`** CLI requires.

Col	Field	Accepted Values	Required
A	Container	The name of the container that will hold the block. Names can be in either short or long format. Short format example: **Dallas**. Long format example: **InControl/Texas/Dallas**. Long format eliminates ambiguity in cases where there are duplicate container names.	Yes
B	IP space	The IP block to create. This should be in the format of a network address (e.g., 10.0.0.0).	Yes
C	Block size	The size of the block in short-notation (e.g., 24 for a 255.255.255.0 network).	Yes
D	Description	A description of the block.	No
E	Block type	The Block Type for the block. If not specified, a block type of **Any** is assumed.	No
F	Block Name	A name for the block. Defaults to system supplied name of *Address space/Block size*.	No
G	Allow Duplicate Space	Whether or not to allow duplicate (overlapping) address space in this block. Accepted values are **true** or **false**. If not specified, defaults to **false**.	No
H	Regional Internet Registry	The Regional Internet Registry this space was obtained from. Accepted values are: **Generic**, **RFC1918**, **ARIN**, **RIPE**, **APNIC**, **LACNIC**, and **AFRINIC**. If not specified, **Generic** is assumed.	No

ExportRootBlock

Col	Field	Accepted Values	Required
I	Organization Id	The organization id for the Regional Internet Registry this space was obtained from. This id must be predefined in IPAM.	No
J	Allocation Reason	The name of a pre-existing Allocation Reason. If Allocation Reason is not currently in IPAM, this field is skipped.	No
K	Allocation Reason Description	A description of the reason for the allocation.	No
L	SWIP/Net Name	SWIP/Net name for the block.	Yes, if required by Container rules
M	Create Reverse Domains	Whether or not to automatically create reverse DNS domain(s) for this block. Accepted values are **true** or **false**. If not specified, defaults to **false**.	No
N	Domain Type	Specify the domain type for the reverse DNS domain(s) to be created when Create Reverse Domains is "true". If not specified, defaults to "Default".	No
O	User Defined Fields	A series of name=value pairs, where the name is the UDF field name/tag and the value is desired value. Multiple pairs can be specified by separating each pair with the '|' character. For example, **UDFone=value one |UDFtwo=value two.** If the UDF type is Checkbox, the valid values are "on" or "off".	Yes, for UDFs defined as required fields.

Deletes

DeleteAddrPool

Overview

The **DeleteAddrPool** CLI allows the user to delete Address Pools in the system. This CLI enables you to specify a file containing a list of address pools to delete.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.DeleteAddrPoolCLI -u <userId> -p <pswd>
-f <delete filename> [-d] [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteAddrPool.sh -u <userId> -p <pswd>
-f <delete filename> [-d] [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DeleteAddrPool.cmd -u <userId> -p <pswd>
-f <delete filename> [-d] [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-d	No	Delete devices within this pool
-f <delete filename>	Yes	The name of the CSV file listing address pools to be deleted. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-deleted) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example deletes the address pools listed in the file *addrpools.txt*.

```
$INCHOME/etc/cli/DeleteAddrPool.sh -u joe -p joepwd -f addrpools.txt
```

File Format

The input file format for **DeleteAddrPool** is the same as for **ImportAddrPool**. The addr pool is identified by its Start Address or Name. Multiple addr pools to be deleted can be specified.

Col	Field	Accepted Values	Required
A	Start Address	The IP Address of the first address in the pool.	Yes
B	End Address	Ignored	No

DeleteAddrPool

Col	Field	Accepted Values	Required
C	Address Pool Type	Ignored	No
D	Name	Address Pool name.	Yes
E	Share Name	Ignored	No
F	Container	Ignored	No
G	Primary Net Service	Ignored	No
H	Failover Net Service	Ignored	No
I	DHCP Option Set	Ignored	No
J	DHCP Policy Set	Ignored	No
K	Allow DHCP Client Classes	Ignored	No
L	Deny DHCP Client Classes	Ignored	No
M	PrefixLength	Ignored	No

DeleteAdmin

Overview

The **DeleteAdmin** CLI allows the user to delete administrators from IPAM. Specify a file containing a list of administrators to delete.

Usage

Direct

```
$INCHOME/jre/bin/java –cp $CLASSPATH
com.diamondip.ipcontrol.cli.DeleteAdminCLI –u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ DeleteAdmin.sh –u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ DeleteAdmin.cmd –u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <delete filename>	Yes	The name of the CSV file listing administrators to be deleted. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-deleted) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.
-v	No	Produces verbose output.

Usage Example

This example deletes the administrators listed in the file *administrators.csv*.

```
$INCHOME/etc/cli/DeleteAdmin.sh –u joe –p joepwd –f administrators.csv
```

File Format

By design, the input file format for **DeleteAdmin** is the same as for **ImportAdmin and ExportAdmin**. Only the administrator login id is required (column A). All other fields are ignored.

DeleteAdminRole

Overview

The **DeleteAdminRole** CLI allows the user to delete administrator roles from IPAM. Specify a file containing a list of administrator roles to delete.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.DeleteAdminRoleCLI -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ DeleteAdminRole.sh -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ DeleteAdminRole.cmd -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <delete filename>	Yes	The name of the CSV file listing administrator roles to be deleted. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-deleted) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.
-v	No	Produces verbose output.

Usage Example

This example deletes the administrator roles listed in the file *adminroles.csv*.

```
$INCHOME/etc/cli/DeleteAdminRole.sh -u joe -p joepwd -f adminroles.csv
```

File Format

By design, the input file format for **DeleteAdminRole** is the same as for **ImportAdminRole and ExportAdminRole**. Only the administrator role name is required (column A). All other fields are ignored.

DeleteAggregateBlock

Overview

The **DeleteAggregateBlock** CLI allows the user to delete an intermediate level Aggregate block from the block hierarchy. By specifying the block to be deleted, IPAM validates and deletes the block. It also adjusts the parent block assignments of any child blocks.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.DeleteAggregateBlockCLI -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteAggregateBlock.sh -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DeleteAggregateBlock.cmd -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <delete filename>	Yes	The name of the CSV file listing blocks to be deleted. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-imported) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example deletes resource records described in the *delaggblocks.csv* file, place into the *delaggblocks.reject* file any records that could not be deleted, and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/DeleteAggregateBlock.sh -u joe -p joepwd -f delaggblocks.csv
-r delaggblocks.reject -e importerrors.txt
```

File Format

Col	Field	Accepted Values	Required
A	Container	The name of the container holding the aggregate block to be deleted. Names can be in either short or long format. Short format example: Dallas. Long format example: InControl/Texas/Dallas. Long format eliminates ambiguity in cases where there are duplicate container names.	Yes
B	Start Address	The start address of the aggregate block.	Yes
C	Block Size	The size of the block in short-notation (e.g., 24 for a 255.255.255.0 network).	Yes
D	Block Type	The Block Type for the block If not specified, a block type of Any is assumed.	No
E	Block Name	A name for the block. Defaults to system supplied name of Address space/Block size.	No
F	Description	A description of the block.	No
G	SWIP Name	SWIP name for the block.	No
H	Allocation Reason	The name of a pre-existing Allocation Reason.	No
I	Allocation Reason Description	A description of the reason for the allocation. Wrap the statement in "quotes" if it contains any commas.	No
J	Interface Name	If this block is being added to a device container, the name of the interface to attach the block to.	No
K	Interface Offset or Address	DEPRECATED. This field will be ignored.	No
L	Create Reverse Domains	Whether or not to automatically create reverse DNS domain(s) for this block. Accepted values are true or false. If not specified, defaults to false.	No
M	Domain Type	Specify the domain type for the reverse DNS domain(s) to be created when Create Reverse Domains is "true". If not specified, defaults to "Default".	No
N	User Defined Fields	A series of name=value pairs, where the name is the UDF name and the value is desired value. Multiple pairs can be specified by separating each pair with the "|" character. For example, UDFone=value one |UDFtwo=value two. If the UDF type is Checkbox, the valid values are "on" or "off".	No
O	Parent Container	The name of the container where the parent block resides.	No
P	Parent Block Address	The address of the parent block	No
Q	Parent Block Size	The size of the parent block in short-notation (e.g., 24 for a 255.255.255.0 network).	No

DeleteBlock

Overview

The **DeleteBlock** CLI allows the user to remove blocks from the system. This CLI allows you to specify a single block to delete on the command line, or to read a file containing a list of blocks to delete.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.DeleteBlockCLI -u <userId> -p <pswd>
[-f <delete filename>] [-r <rejects file>] [-e <error messages>] [-b <block Name>]
[-c <container name>] [-x] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteBlock.sh -u <userId> -p <pswd>
[-f <delete filename>] [-r <rejects file>] [-e <error messages>] [-b <block Name>]
[-c <container name>] [-x] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DeleteBlock.cmd -u <userId> -p <pswd>
[-f <delete filename>] [-r <rejects file>] [-e <error messages>] [-b <block Name>]
[-c <container name>] [-x] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-x	No	When presents indicates the CSV file specified by the –f flag is from the export*Block cli, where the file format lists the container name as the first column and the block name as the 4th column. When the –x flag is not present, the file format is blockname, containername.
-f <delete filename>	Either –f or –b must be specified.	The name of the CSV file listing blocks to be deleted. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-deleted) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.
-b <block name>	Either –f or –b must be specified.	The name of the block to be deleted. This is typically the CIDR address, e.g., 10.1.2.0/24. However, if the block name was set manually during allocation, that value must be used instead.
-c <container name>	Must be specified if the block name is not unique.	The name of the container holding the block. If this parameter is supplied, the container is searched for the block to delete. Otherwise, the whole system is searched.

Usage Example

This example deletes the block `10.1.2.0/24` from the system. As mentioned above, this example assumes that there is only one block with this name in the system.

```
$INCHOME/etc/cli/DeleteBlock.sh -u joe -p joepwd -b 10.1.2.0/24
```

File Format

Col	Field	Accepted Values	Required
A	Block Name	The name of the block to delete. This is typically the CIDR address, e.g. 10.1.2.0/24. However, if the block name was set manually during allocation, that value must be used instead.	Yes
B	Container Name	The name of the container holding the block.	No

DeleteContainer

Overview

The **DeleteContainer** CLI allows the user to remove individual containers from the system. This CLI allows you to specify a file containing a list of containers to delete. Only containers that contain no blocks, services or child containers are available for deletion.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.DeleteContainerCLI -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteContainer.sh -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DeleteContainer.cmd -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <delete filename>	Yes	The name of the CSV file listing containers to be deleted. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-deleted) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example deletes the containers listed in the file *containers.txt*.

```
$INCHOME/etc/cli/DeleteContainer.sh -u joe -p joepwd -f containers.txt
```

File Format

By design, the input file format for **DeleteContainer** is the same as for **ImportContainer**. This way, one can use the same file to delete a set of containers and then re-add them without copying data. Only the container name is required.

Col	Field	Accepted Values	Required
A	Container Name	The name of the container. The name can be either qualified or unqualified. An unqualified name must be unique. A qualified name must start with the root container and include the complete container path to the desired container. The container names should be separated by slashes.	Yes
B	Container Description	Ignored	No
C	Parent Container	Ignored	No
D	Container Type	Ignored	No
E	Rule1	Ignored	No
F	Rule2	Ignored	No
G	Rule3	Ignored	No
H	Rule4	Ignored	No
I	Rule5	Ignored	No
J	Information Template	Ignored	No
K	User Defined Fields	Ignored	No

DeleteDevice

Overview

The **DeleteDevice** CLI allows the user to remove individual devices from the system. This CLI allows you to specify a file containing a list of devices to delete.

Note that this is not used to delete devices of type "Interface". Use the ImportChildBlock CLI with the overwrite parameter to delete Interface-type devices.

Usage

Direct

```
$INCHOME/jre/bin/java –cp $CLASSPATH
com.diamondip.ipcontrol.cli.DeleteDeviceCLI –u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteDevice.sh –u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DeleteDevice.cmd –u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <delete filename>	Yes	The name of the CSV file listing devices to be deleted. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-deleted) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example deletes the devices listed in the file *devices.txt*.

```
$INCHOME/etc/cli/DeleteDevice.sh –u joe –p joepwd –f devices.txt
```

File Format

By design, the input file format for **DeleteDevice** is the same as for **ImportDevice**. This way, one can use the same file to the delete a set of devices and then re-add them without copying data. In the simplest case, only the IP Address is required. The container might also be required if overlapping address space is in use and the device is in a block that is part of that overlapping space. In this case, the Container is required to uniquely determine the device.

Col	Field	Accepted Values	Required
A	IP Address	The IP Address of the Device	Yes
B	Address Type	Ignored	No
C	Host Name	Ignored	No
D	Device Type	Ignored	No
E	Hardware Type	Ignored	No
F	MAC Address	Ignored	No
G	Resource Record Flag	Ignored	No
H	Domain Name	Ignored	No
I	Container	The name of the container holding the block to which the device belongs.	Yes, if overlapping space is in use and the block in which the device resides is ambiguous.

DeleteDomain

Overview

The **DeleteDomain** CLI allows the user to remove individual domains from the system. This CLI allows you to specify a file containing a list of domains to delete.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.DeleteDomainCLI -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteDomain.sh -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DeleteDomain.cmd -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <delete filename>	Yes	The name of the CSV file listing domains to be deleted. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-deleted) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.
-v <verbose>	No	Produces verbose output.

Usage Example

This example deletes the domains listed in the file *domains.txt*.

```
$INCHOME/etc/cli/DeleteDomain.sh -u joe -p joepwd -f domains.txt
```

File Format

By design, the input file format for **`DeleteDomain`** is the same as for **`ImportDomain`**. This way, one can use the same file to delete a set of domains and then re-add them without copying data. In the simplest case, only the Domain Name is required. The Domain Type might also be required if it is not "Default".

Col	Field	Accepted Values	Required
A	Domain Name	The name of the domain being created.	Yes
B	Domain Type	Domain type name already defined to IPAM. If not specified, the "Default" domain type will be used.	Yes, when not "Default".
C	Managed	Ignored	No
D	Delegated	Ignored	No
E	Reverse	Ignored	No
F	Derivative	Ignored	No
G	Template Domain	Ignored	No
H	Serial Number	Ignored	No
I	Refresh	Ignored	No
J	Retry	Ignored	No
K	Expire	Ignored	No
L	Negative Cache TTL	Ignored	No
M	Default TTL	Ignored	No
N	Contact	Ignored	No
O	Information Template	Ignored	No
P	User Defined Fields	Ignored	No

DeleteDeviceInterface

Overview

The **DeleteDeviceInterface** CLI allows the user to remove device interfaces from multi-homed devices in the system. This CLI enables you to specify a file containing a list of device interfaces to delete. To remove all interfaces, use the **DeleteDevice** CLI.

Note that this is not used to delete devices of type "Interface". Use the ImportChildBlock CLI with the overwrite parameter to delete Interface-type devices.

You can also use the **ImportDevice** CLI with the overwrite option to add and remove device interfaces.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.DeleteDeviceInterfaceCLI -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteDeviceInterface.sh -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DeleteDeviceInterface.cmd -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <delete filename>	Yes	The name of the CSV file listing device interfaces to be deleted. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-deleted) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example deletes the device interfaces listed in the file *deviceInterfaces.txt*.

```
$INCHOME/etc/cli/DeleteDeviceInterface.sh -u joe -p joepwd -f deviceInterfaces.txt
```

File Format

The input file format for **`DeleteDeviceInterface`** is the same as for **`ExportDevice`**. The device is identified by its IP Address. The container must also be specified if the device's IP Address is not unique due to overlapping address space. The interface is identified by its name. Multiple interfaces to be deleted can be specified.

Col	Field	Accepted Values	Required
A	IP Address	The IP address of the device. If multiple interfaces are to be deleted, you may specify their IP addresses delimited by '|' (as output by ExportDevice). However, only 1 IP address is required to identify the device.	Yes
B	Address Type	Ignored	No
C	Host Name	Ignored	No
D	Device Type	Ignored	No
E	Hardware Type	Ignored	No
F	MAC Address	Ignored	No
G	Resource Record Flag	Ignored	No
H	Domain Name	Ignored	No
I	Container	The name of the container holding the block to which the device belongs. If multiple containers are required to resolve multiple IP addresses specified in column A, multiple containers may be specified, delimited by '|'.	Yes, if overlapping space is in use and the block in which the device resides is ambiguous.
J	Domain Type	Ignored	No
K	Description	Ignored	No
L	User Defined Fields	Ignored	No
M	Aliases	Ignored	No
N	Ignore Warning	Ignored	No
O	Interface Names	Specify the names of the interfaces to be deleted, delimited by '|'. If multiple IP addresses were specified in column A, the number of interfaces specified must match the number of IP Addresses specified.	Yes
P	Exclude from Discovery Flags	Ignored	No

DeleteDeviceResourceRecord

Overview

The **DeleteDeviceResourceRecord** CLI allows the user to delete DNS resource records for a device from IPAM.

Usage

Direct

```
$INCHOME/jre/bin/java –cp $CLASSPATH
com.diamondip.ipcontrol.cli.DeleteDeviceResourceRecordCLI –u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteDeviceResourceRecord.sh –u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DeleteDeviceResourceRecord.cmd –u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <delete filename>	Yes	The name of the CSV file describing records to be deleted. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-deleted) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example deletes resource records described in the *delresourcerecs.csv* file, places into the *delresourcerecs.reject* file any records that could not be deleted, and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/DeleteDeviceResourceRecord.sh –u joe –p joepwd –f delresourcerecs.csv
–r delresroucerecs.reject –e importerrors.txt
```

File Format

Col	Field	Accepted Values	Required
A	Domain	The name of the domain for this resource record.	Yes
B	DomainType	The domain type of the domain. Defaults to "Default"	No
C	Owner	The OWNER section of the resource record.	Yes
D	Host Name	The device host name.	Yes, unless IP Address is specified.
E	IP Address	The IP Address of the Device.	Yes, unless Host Name is specified.
F	Container	The name of the container that holds the device. This is required only if there is overlapping address space in use and the ip address is in overlapping space. The container is then used to uniquely determine the device.	Yes, if IP Address in overlap-ping space.
G	TTL	The Time To Live.	No
H	Class	The value currently supported is **IN**. If not specified, defaults to **IN**.	No
I	Resource Record Type	The type of resource record being deleted.	Yes
J	Data	The text for the DATA area of the resource record.	Yes

DeleteDomainResourceRecord

Overview

The **DeleteDomainResourceRecord** CLI allows the user to delete DNS resource records for a DNS domain from IPAM.

Note: For Glue records that link one zone to another, use the **DeleteZoneResourceRecord** CLI.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.DeleteDomainResourceRecordCLI -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteDomainResourceRecord.sh -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DeleteDomainResourceRecord.cmd -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <delete filename>	Yes	The name of the CSV file describing records to be deleted. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-deleted) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example deletes resource records described in the *delresourcerecs.csv* file, places into the *delresourcerecs.reject* file any records that could not be deleted, and reports errors to the *deleteerrors.txt* file.

```
$INCHOME/etc/cli/DeleteDomainResourceRecord.sh -u joe -p joepwd -f delresourcerecs.csv
-r delresroucerecs.reject -e deleteerrors.txt
```

File Format

Col	Field	Accepted Values	Required
A	Domain	The name of the domain for this resource record.	Yes
B	DomainType	The domain type of the domain. Defaults to "Default".	No
C	Owner	The OWNER section of the resource record.	Yes
D	TTL	The Time To Live.	No
E	Class	The value currently supported is **IN**. If not specified, defaults to **IN**.	No
F	Resource Record Type	The type of resource record being deleted.	Yes
G	Data	The text for the DATA area of the resource record.	Yes

DeleteNetElement

Overview

The **DeleteNetElement** CLI enables the user to remove individual network elements from the system. This CLI allows you to specify a file containing a list of network elements to delete.

Note: This CLI has been deprecated as of IPAM 8.1.2. Use the **DeleteNetworkElement** CLI instead. ImportNetElement support will be removed in a future release.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.DeleteNetElementCLI -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteNetElement.sh -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DeleteNetElement.cmd -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <delete filename>	Yes	The name of the CSV listing network elements to be deleted See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-deleted) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example deletes network elements listed in the *netelements.csv* file, places into the *netelements.reject* file any records that could not be deleted, and reports errors to the *deleteerrors.txt* file.

```
$INCHOME/etc/cli/DeleteNetElement.sh -u joe -p joepwd -f netelements.csv
-r netelements.reject -e deleteerrors.txt
```

File Format

By design, the input file format for **`DeleteNetElement`** is the same as the format for **`ImportNetElement`** and **`ExportNetElement`**. This way, one can use the same file to delete a set of network elements and then re-add them without copying data. Specify the name or the IP address of the network element. If the IP address is not unique, the name must be specified.

Col	Field	Accepted Values	Required
A	Name	The name of the Network Element.	Yes, unless a unique IP address is specified
B	IP Address/FQDN	The IP address or fully-qualified domain name (FQDN) of the Network Element.	Yes, unless the name is specified
C	Vendor	Ignored	No
D	Model	Ignored	No
E	Type	Ignored	No
F	Global Sync	Ignored	No
G	Agent Name	Ignored	No
H	Telnet User	Ignored	No
I	Telnet Password	Ignored	No
J	Enable Password	Ignored	No
K	Read Community String	Ignored	No
L	Interface List	Ignored	No
M	V3 Username	Ignored	No
N	V3 Authentication Protocol	Ignored	No
O	V3 Authentication Password	Ignored	No
P	V3 Privacy Protocol	Ignored	No
Q	V3 Privacy Password	Ignored	No
R	V3 Context Name	Ignored	No
S	V3 Engine ID	Ignored	No

DeleteNetElementInterface

Overview

The **DeleteNetElementInterface** CLI allows the user to delete network element interfaces from IPAM. This CLI allows you to specify a file containing a list of network element interfaces to delete.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.DeleteNetElementInterfaceCLI -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteNetElementInterface.sh -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DeleteNetElementInterface.cmd -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <delete filename>	Yes	The name of the CSV file listing the interfaces to be deleted. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (not deleted) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.
-v	No	Produces verbose output.

Usage Example

This example deletes Network Elements interfaces listed in the *netelementinterfaces.csv* file, places into the *netelementinterfaces.reject* file any records that could not be deleted, and reports errors to the *deleteerrors.txt* file.

```
$INCHOME/etc/cli/DeleteNetElementInterface.sh -u joe -p joepwd
-f netelementinterfaces.csv -r netelementinterfaces.reject -e deleteerrors.txt
```

File Format

By design, the input file format for **DeleteNetElementInterface** is the same as for **ImportNetElementInterface**. This way, one can use the same file to the delete a set of interfaces and then re-add them without copying data. Only the network element name and interface name are required.

Col	Field	Accepted Values	Required
A	Name	The name of a Network Element already defined in IPAM.	Yes
B	Interface Name	The name of the interface to delete.	Yes
C	Status	The status of the interface. This is not used for delete.	No

DeleteNetService

Overview

The **DeleteNetService** CLI enables the user to remove individual network services from the system. This CLI allows you to specify a file containing a list of network services to delete.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.DeleteNetServiceCLI -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteNetService.sh -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DeleteNetService.cmd -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <delete filename>	Yes	The name of the CSV listing of network services to be deleted See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-deleted) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example deletes network services listed in the *netservices.csv* file, places into the *netservices.reject* file any records that could not be deleted, and reports errors to the *deleteerrors.txt* file.

```
$INCHOME/etc/cli/DeleteNetService.sh -u joe -p joepwd -f netservices.csv
-r netservices.reject -e deleteerrors.txt
```

File Format

By design, the input file format for **DeleteNetService** is the same as the format for **ImportNetService** and **ExportNetService**. This way, one can use the same file to delete a set of network services and then re-add them without copying data. Specify the name and the type of the network service. If the type is not specified, it defaults to "DHCP".

Col	Field	Accepted Values	Required
A	Name	The name of the Network Service. This can be any combination of letters and numbers.	Yes
B	IP Address/FQDN	Ignored	No
C	Type	The type of Network Service. Accepted value is **dhcp**. If this column is left blank, **dhcp** is assumed.	No
D	Product name	Ignored	No
E	Agent name	Ignored	No
F	Global Sync	Ignored	No
G	Collection Method	Ignored	No
H	User name for collection	Ignored	No
I	Password for collection	Ignored	No
J	Collection port	Ignored	No
K	Container(s)	Ignored	
L	VendorInfo	Ignored	No
M	WarningThreshold	Ignored	No

DeleteNetworkElement

Overview

The **DeleteNetworkElement** CLI enables the user to remove individual network elements from the system. This CLI allows you to specify a file containing a list of network elements to delete. It deprecates DeleteNetElement CLI. While the two methods are different in name only, DeleteNetworkElement is being added for tight name alignment with its Import and Export CLI counterparts.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.DeleteNetworkElementCLI -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteNetworkElement.sh -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DeleteNetworkElement.cmd -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <delete filename>	Yes	The name of the CSV listing network elements to be deleted See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-deleted) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example deletes network elements listed in the *networkelements.csv* file, places into the *networkelements.reject* file any records that could not be deleted, and reports errors to the *deleteerrors.txt* file.

```
$INCHOME/etc/cli/DeleteNetworkElement.sh -u joe -p joepwd -f networkelements.csv
-r networkelements.reject -e deleteerrors.txt
```

File Format

By design, the input file format for **`DeleteNetworkElement`** is the same as the format for **`ImportNetworkElement`** and **`ExportNetworkElement`**. This way, one can use the same file to delete a set of network elements and then re-add them without copying data. Specify the name or the IP address of the network element. If the IP address is not unique, the name must be specified.

Col	Field	Accepted Values	Required
A	Name	The name of the Network Element.	Yes, unless a unique IP address is specified
B	Description	Ignored	No
C	IP Address	The IP address of the Network Element.	Yes, unless the name is specified
D	Type	Ignored	No
E	Vendor	Ignored	No
F	Model	Ignored	No
G	Agent Name(s)	Ignored	No
H	Global Sync	Ignored	No
I	SNMP Version	Ignored	No
J	Read Community String	Ignored	No
K	SNMP Timeout	Ignored	No
L	SNMP Retries	Ignored	No
M	V3 Username	Ignored	No
N	V3 Authentication Protocol	Ignored	No
O	V3 Authentication Password	Ignored	No
P	V3 Privacy Protocol	Ignored	No
Q	V3 Privacy Password	Ignored	No
R	V3 Context Name	Ignored	No
S	V3 Engine ID	Ignored	No
T	Interface List	Ignored	No
U	Device Interface Template and interface status	Ignored	No
V	SNMP sysName	Ignored	No
W	SNMP sysDescr	Ignored	No
X	SNMP sysLocation	Ignored	No
Y	SNMP sysServices	Ignored	No

DeleteNetworkLink

Overview

The **DeleteNetworkLink** CLI allows the user to delete logical Network Links from IPAM. Specify a file containing a list of network links to delete.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.DeleteNetworkLinkCLI -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ DeleteNetworkLink.sh -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ DeleteNetworkLink.cmd -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <delete filename>	Yes	The name of the CSV file listing address pools to be deleted. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-deleted) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example deletes the network links listed in the file *networklinks.txt*.

```
$INCHOME/etc/cli/DeleteNetworkLink.sh -u joe -p joepwd -f networklinks.txt
```

File Format

By design, the input file format for **DeleteNetworkLink** is the same as for **ImportNetworkLink**. This way, one can use the same file to delete a set of network links and then re-add them without copying data. Only the Network Link Name is required. All other fields are ignored.

DeletePrefixPool

Overview

The **DeletePrefixPool** CLI allows the user to delete Prefix Pools in the system. This CLI enables you to specify a file containing a list of prefix pools to delete.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.DeletePrefixPoolCLI -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DeletePrefixPool.sh -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DeletePrefixPool.cmd -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <deletet filename>	Yes	The name of the CSV file listing prefix pools to be deleted. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-deleted) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example deletes the prefix pools listed in the file *prefixpools.txt*.

```
$INCHOME/etc/cli/DeletePrefixPool.sh -u joe -p joepwd -f prefixpools.txt
```

File Format

The input file format for **DeletePrefixPool** is the same as for **ImportPrefixPool**. The prefix pool is identified by its Start Address or Name. Multiple prefix pools to be deleted can be specified.

Col	Field	Accepted Values	Required
A	Start Addr	The Start address of the prefix pool.	Yes
B	Length	Ignored	No
C	Prefix Pool Type	Ignored	No
D	Name	Name of the prefix pool.	Yes
E	Container	Ignored	No
F	Delegated Prefix Length	Ignored	No
G	Longest Prefix Length	Ignored	No
H	Shortest Prefix Length	Ignored	No
I	Primary Net Service	Ignored	No
J	DHCP Option Set	Ignored	No
K	DHCP Policy Set	Ignored	No
L	Allow DHCP Client Classes	Ignored	No
M	Deny DHCP Client Classes	Ignored	No
N	Overlap Interface IP	Ignored	No

DeleteTask

Overview

The **DeleteTask** CLI allows the user to delete tasks from the system.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.DeleteTaskCLI -u <userId> -p <pswd>
[-t <idlist> | -r <retain> | -d <date>] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteTask.sh -u <userId> -p <pswd>
[-t <idlist> | -r <retain> | -d <date>] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DeleteTask.cmd -u <userId> -p <pswd>
[-t <idlist> | -r <retain> | -d <date>] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-t <idlist>	One of –t, -r, or –d must be specified.	One or more task IDs to delete. Multiple tasks are separated by commas with no spaces. In Windows, enclose a list of tasks in quotes, for example, -t "123,456".
-r <retain>	One of –t, -r, or –d must be specified.	The number of days of tasks to retain. For example, if –r 30 is specified, tasks older than 30 days will be deleted.
-d <date>	One of –t, -r, or –d must be specified.	Tasks older than this date will be deleted, in the format yyyy/mm/dd. For example, for a date of 2005/12/31, all tasks prior to that date will be deleted.
-v	No	Verbose option. The CLI will print out more information about the request. In particular, it will print how many tasks were deleted.

Usage Example

This example deletes tasks older than 60 days.

```
$INCHOME/etc/cli/DeleteTask.sh -u joe -p joepwd -r 60 -v
```

File Format

None. This CLI uses command line arguments only.

DeleteZone

Overview

The **DeleteZone** CLI enables the user to remove zones from the system. This CLI allows you to specify a file containing a list of zones to delete.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.DeleteZoneCLI -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteZone.sh -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DeleteZone.cmd -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <delete filename>	Yes	The name of the CSV listing zones to be deleted See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-deleted) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.
-v <verbose>	No	Produces verbose output.

Usage Example

This example deletes zones listed in the *zones.csv* file, places into the *zones.reject* file any records that could not be deleted, and reports errors to the *deleteerrors.txt* file.

```
$INCHOME/etc/cli/DeleteZone.sh -u joe -p joepwd -f zones.csv
-r zones.reject -e deleteerrors.txt
```

File Format

By design, the input file format for **`DeleteZone`** is the same as the format for **`ImportZone`**. This way, one can use the same file to delete a set of zones and then re-add them without copying data.

Col	Field	Accepted Values	Required
A	Server	The network service name of the DNS Server.	Yes
B	Zone type	Ignored	No
C	Domain Name	Domain name already defined to IPAM.	Yes
D	Domain Type	Domain type name already defined to IPAM. If not specified, the "Default" domain type will be used.	No
E	Update Zone	Ignored	No
F	View	The name of the view containing the zone to be deleted. If not specified, 'Default.' will be used.	No
G	Filename	Ignored	No
H	Automatic Generation of NS/GLUE Records	Ignored	No
I	MNAME	Ignored	No
J	Allow Zone Transfers to DNS Listener	Ignored	No
K	Masters	Ignored	No
L	Zone extensions Prior	Ignored	No
M	Zone extensions After	Ignored	No
N	Template Zone	Ignored	No
O	Alias Zone	Ignored	No
P	Galaxy Name	Ignored	No
Q	Allow Update ACL	Ignored	No
R	Update Policy	Ignored	No
S	Dynamic Zone	Ignored	No
T	Publish NS	Ignored	No

DeleteZoneResourceRecord

Overview

The **DeleteZoneResourceRecord** CLI allows the user to delete DNS resource records for a zone from IPAM. These are the Glue records that link one zone to another.

Note: This interface should not be confused with the **DeleteDomainResourceRecord** CLI, which is used to delete records from a **domain**.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.DeleteZoneResourceRecordCLI -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteZoneResourceRecord.sh -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DeleteZoneResourceRecord.cmd -u <userId> -p <pswd>
-f <delete filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the CSV file describing records to be deleted. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-deleted) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

This example deletes resource records described in the *delresourcerecs.csv* file, places into the *delresourcerecs.reject* file any records that could not be deleted, and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/DeleteZoneResourceRecord.sh -u joe -p joepwd -f delresourcerecs.csv
-r delresroucerecs.reject -e importerrors.txt
```

File Format

Col	Field	Accepted Values	Required
A	Server	The network service name of the DNS Server.	Yes
B	View	The name of the view for this zone. If supplied the view must exist. If not specified, 'Default' will be used.	Yes
C	Zone	The name of the zone, which is the top level domain name.	Yes
D	Owner	The OWNER section of the resource record.	Yes
E	TTL	The Time To Live. If specified, must match the value in the record to be deleted.	No
F	Class	The value currently supported is **IN**. If not specified, defaults to **IN**.	No
G	Resource Record Type	The type of resource record being deleted.	Yes
H	Data	The text for the DATA area of the resource record.	Yes

Gets

GetContainerParentHierarchy

Overview

The **GetContainerParentHierarchy** CLI allows the user to retrieve a fully qualified list of all parents of a named container. This will typically return a single parent listing for logical containers and a multiple parent listing for device containers.

Usage

Direct

```
$INCHOME/jre/bin/java –cp $CLASSPATH
com.diamondip.ipcontrol.cli.GetContainerParentHierarchyCLI –u <userId> -p <pswd>
-c <container name> [-f <output filename>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/GetContainerParentHierarchy.sh –u <userId> -p <pswd>
-c <container name> [-f <output filename>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/GetContainerParentHierarchy.cmd –u <userId> -p <pswd>
-c <container name> [-f <output filename>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-c	Yes	Container name to return parent hierarchy for.
-f <output filename>	No	The name of a CSV file to write container parent hierarchy to. If omitted output is written to stdout.
-e <error messages>	No	The name of the file that error messages will be reported in.
-v <verbose>	No	Produces verbose output.

Usage Example

This example writes the parent hierarchy of device container "router" to stdout.

```
$INCHOME/etc/cli/ GetContainerParentHierarchy.sh –u joe –p joepwd –c router
```

File Format

N/A

GetEffectiveDhcpServersForContainer

Overview

The **GetEffectiveDhcpServersForContainer** CLI allows the user to retrieve a list of all valid and accessible DHCP servers for a given container.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.GetEffectiveDhcpServersForContainerCLI -u <userId> -p <pswd>
-c <container name> [-f <output filename>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/GetEffectiveDhcpServersForContainer.sh -u <userId> -p <pswd>
-c <container name> [-f <output filename>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/GetEffectiveDhcpServersForContainer.cmd -u <userId> -p <pswd>
-c <container name> [-f <output filename>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-c	Yes	Container name to return effective DHCP servers for.
-f <output filename>	No	The name of the file to write the data to. If no file name is specified, results are outputted to the screen.
-e <error messages>	No	The name of the file that error messages will be reported in.
-v <verbose>	No	Produces verbose output.

Usage Example

This example lists the effective DHCP servers for the container "Malvern" in the file "effectiveServers.csv" in the format shown in the File Format section..

$INCHOME/etc/cli/ GetEffectiveDhcpServersForContainer.sh –u joe –p joepwd –c Malvern –f effectiveServers.csv

File Format

Col	Field	Description
A	Name	The name of the DHCP Server.
B	IP Address	The IP Address of the DHCP Server.
C	Product	The product name of the DHCP Server.
D	Agent	The IPAM agent that manages the server

Col	Field	Description
E	Default Threshold	The default alert threshold applied to all scopes managed by this server. This must be a number between 0 and 100. Defaults to 90.
F	Global Sync	TRUE indicates this server is included in Global Sync tasks. Defaults to false.
G	Configuration Path	The path to the server's configuration file.
H	Lease Path	The path to the lease file.
I	Start Script	The path to the script that starts the server.
J	Stop Script	The path to the script that stops the server.
K	Collection Type	SCP or FTP
L	Collection Port	Must be between 1 and 65535. Defaults to 21 for FTP and 22 for SCP.
M	Collection User	User name for SCP/FTP access to the executive.
N	Collection Password	Password for the collection user.
O	Collect Backup Subnets	TRUE indicates statistics on backup subnets will be collected. Defaults to FALSE.
P	CLI Command	Collection program name
Q	CLI User	Collection program user credential.
R	CLI Password	Collection program password credential
S	CLI Arguments	Arguments to the Collection program. Differs according to product.
T	DDNS	Specify 'interim', 'standard', or 'lastin' to enable dynamic DNS updates when this server issues a lease. Defaults to 'none'.
U	DHCP Option Set	The name of an option set defined in IPAM.
V	DHCP Policy Set	The name of a policy set defined in IPAM.
W	DHCP Client Classes	The names of client classes defined in IPAM that this server will be using. Multiple client classes are separated with a vertical bar ("|").
X	DHCP Failover IP Address	The IP Address used by the DHCP server for failover communications.
Y	DHCP Failover Port	The Port used by the DHCP server for failover communications.
Z	Configuration File Pre-Extension	Text to prepend to the DHCP server configuration file. This can be the text itself, or a reference to a file. If the field begins with "file:", then the remainder of the field is treated as a file name and the file's contents are used.
AA	Configuration File Post-Extension	Text to append to the DHCP server configuration file. This can be the text itself, or a reference to a file. If the field begins with "file:", then the remainder of the field is treated as a file name and the file's contents are used.
AB	V4V6Both	IP version supported by the DHCP server. One of 'V4', 'V6' or 'Both'

Tasks

ArpDiscoverNetElement Task

Overview

The ArpDiscoverNetElement CLI allows the user to create an ARP Discover task on a given Net Element.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.ArpDiscoverNetElementCLI -u <userId> -p <pswd>
[-n <Net Element name> | -i <Net Element IP address>]
[--performnethostadds][--updatereclaim][--ignoredups] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ArpDiscoverNetElementCLI.sh -u <userId> -p <pswd>
[-n <Net Element name> | -i <Net Element name>]
[--performnethostadds] [--updatereclaim] [--ignoredups] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ArpDiscoverNetElementCLI.cmd -u <userId> -p <pswd>
[-n <Net Element name> | -i <Net Element name>]
[--performnethostadds] [--updatereclaim] [--ignoredups] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-i <IP Address>	Yes, unless –n specified	IP Address of the Net Element to discover.
-n <Name>	Yes, unless –i specified	Name of the Net Element to discover.
--performnethostadds	No	If set, the task will add new hosts found during discovery.
--updatereclaim	No	If set, the task will update the last discovered counters for blocks and hosts defined within the network element.
--ignoredups	No	If set, the task will ignore duplicate hostname errors when adding new hosts.
-reverselookup	No	If set, the task will do a reverse lookup of the address in order to attempt to discover the hostname.
-importinvalid	No	If set, ignore the invalid flag for this entry on the router and import it anyway.
-v	No	Produces verbose output, providing the task id.

Usage Example

This creates a task to perform an ARP discover on a Net Element with IP Address 10.40.23.6, and to perform net host adds as part of the processing.

```
$INCHOME/etc/cli/ArpDiscoveryNetElementCLI.sh –u joe –p joepwd –i 10.40.23.6
--performnethostadds
```

Return codes

If the task is scheduled successfully, an errorlevel of 0 is set. If –v is specified, the task number is printed to the screen. That task number can then be passed to the **`TaskStatus`** CLI to obtain the status of that task.

If the task is NOT scheduled successfully, the errorlevel returned is a non-zero number, and an error message is printed to the console.

DhcpConfigurationTask

Overview

The **DhcpConfigurationTask** CLI allows the user to create a DHCP Configuration task. DHCP Configuration tasks generate and deploy configuration files for DHCP servers.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.DhcpConfigurationTaskCLI -u <userId> -p <pswd>
[-n <DHCP server name> | -i <DHCP server IP address>] [-a] [-c] [-f] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DhcpConfigurationTask.sh -u <userId> -p <pswd>
[-n <DHCP server name> | -i <DHCP server IP address>] [-a] [-c] [-f] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DhcpConfigurationTask.cmd -u <userId> -p <pswd>
[-n <DHCP server name> | -i <DHCP server IP address>] [-a] [-c] [-f] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-i <IP Address>	Yes, unless –n specified	IP Address of the Network Service (DHCP Server) to configure.
-n <Name>	Yes, unless –i specified	Name of Network Service (DHCP server) to configure. If an IPV4 and IPV6 server exist with the same name, then use the –i parameter to identify the server by the address.
-a	No	If set, the task will stop if errors are detected during the configuration generation.
-c	No	If set, the task will not execute a push unless the DHCP server configuration has changed.
-f	No	If set, any associated failover servers will also have their configurations generated and deployed.
-v	No	Produces verbose output, providing the task id.

Usage Example

This creates a task to perform a DHCP configuration deployment for the `dhcp5` DHCP server. In addition, if there are any failover servers associated with `dhcp5`, their configurations are also deployed. Lastly, if there are any errors, the task will abort.

```
$INCHOME/etc/cli/DhcpConfigurationTask.sh -u joe -p joepwd -n dhcp5 -a -f
```

Return codes

If the task is scheduled successfully, an errorlevel of 0 is passed, along with a string including the task number. That task number can then be passed to the **`TaskStatus`** CLI to obtain the status of that task.

If the task is NOT scheduled successfully, the errorlevel returned is a negative number, and an error message is printed to the console.

DHCPUtilization

Overview

The **DHCPUtilization** CLI allows the user to issue an immediate DHCP Collection task to collect statistics on the utilization of a DHCP server.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.DHCPUtilizationCLI -u <userId> -p <pswd>
[-n <network service name> | -i <network service ip address>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DHCPUtilization.sh -u <userId> -p <pswd>
[-n <network service name> | -i <network service ip address>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DHCPUtilization.cmd -u <userId> -p <pswd>
[-n <network service name> | -i <network service ip address>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-i <IP Address>	Yes	IP Address of the device to discover. Required only if Network Element Name not specified.
-n <Name>	Yes	Name of Network Service (DHCP server) to discover. Required only if IP Address not specified.
-v	No	Produces verbose output, providing the task id.

Usage Example

This example creates a task on the queue to perform a DHCP utilization collection for the `dhcp5` DHCP server.

```
$INCHOME/etc/cli/DHCPUtilization.sh -u joe -p joepwd -n dhcp5
```

Return codes

If the task is scheduled successfully, an errorlevel of 0 is passed, along with a string including the task number. That task number can then be passed to the **TaskStatus** CLI to obtain the status of that task.

If the task is NOT scheduled successfully, the errorlevel returned will be a negative number, and an error message will be printed to the console.

DiscoverNetElement

Overview

The **`DiscoverNetElement`** CLI allows the user to issue an immediate Discover task to discover the interfaces bound to a network element already defined in IPAM.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.DiscoverNetElementCLI -u <userId> -p <pswd>
[-n <network element name> | -i <netelement address>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DiscoverNetElement.sh -u <userId> -p <pswd>
[-n <network element name> | -i <netelement address>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DiscoverNetElement.cmd -u <userId> -p <pswd>
[-n <network element name> | -i <netelement address>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-i <IP Address | FQDN>	Yes	IP Address or fully-qualified domain name (FQDN) of the device to discover. Required only if Network Element Name not specified.
-n <Name>	Yes	Name of Network Element (device) to discover. Required only if IP Address or FQDN not specified.
-v	No	Produces verbose output, providing the task id.

Usage Examples

This example creates a task on the queue to perform a Discover for the device found at the IP address `10.10.23.4`.

```
$INCHOME/etc/cli/DiscoverNetElement.sh -u joe -p joepwd -i 10.10.23.4
```

This example creates a task to perform a Discover for the device with the DNS name `corerouter1.mycompany.com`.

```
$INCHOME/etc/cli/DiscoverNetElement.sh -u joe -p joepwd -i corerouter1.mycompany.com
```

This example creates a task to perform a Discover for the IPAM network element named `router3`.

```
$INCHOME/etc/cli/DiscoverNetElement.sh -u joe -p joepwd -n router3
```

Return codes

If the task is scheduled successfully, an errorlevel of 0 is passed, along with a string including the task number. That task number can then be passed to the **`TaskStatus`** CLI to obtain the status of that task.

If the task is NOT scheduled successfully, the errorlevel returned will be a negative number, and an error message will be printed to the console.

DnsConfigurationTask

Overview

The **DnsConfigurationTask** CLI allows the user to create a DNS Configuration task. DNS Configuration tasks generate and deploy configuration and zone files for DNS servers.

Usage

Direct

```
$INCHOME/jre/bin/java –cp $CLASSPATH
com.diamondip.ipcontrol.cli.DnsConfigurationTaskCLI –u <userId> -p <pswd>
[-n <Name> | -i <IP address>] [-a] [-c] [-d] [-g] [-k] [-s] [-w <View>] [-z <Zone>]
[-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DnsConfigurationTask.sh –u <userId> -p <pswd>
[-n <Name> | -i <IP address>] [-a] [-c] [-d] [-g] [-k] [-s] [-w <View>] [-z <Zone>]
[-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DnsConfigurationTask.cmd –u <userId> -p <pswd>
[-n <Name> | -i <IP address>] [-a] [-c] [-d] [-g] [-k] [-s] [-w <View>] [-z <Zone>]
[-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-i <IP Address>	Yes, unless –n specified	IP Address of the DNS Server to configure.
-n <Name>	Yes, unless –i specified	Name of DNS server to configure.
-a	No	By default, the task will abort if errors are detected during the configuration generation. Set this option to continue with the deployment in spite of errors.
-c	No	Changes only. With –d, creates a task to send only changed resource records. Without –d, creates a task to push only changed zones.
-d	No	Use Dynamic DNS instead of recreating configuration and zone files. With –c, only changed resource records are sent. Without –c, all resource records are sent.
-g	No	By default, the generated configuration file is checked for errors. Specify this option to suppress that check.
-k	No	By default, the generated zone files are checked for errors. Specify this option to suppress that check.
-s	No	Sends only resource records created in IPAM. Dynamic DNS (-d) must be specified. Use this option to periodically refresh the records in Microsoft AD DNS to prevent their scavenging, while not interfering with the intended scavenging of dynamic records.

Parameter	Required	Description
-v	No	Specify this option to receive an output message to the console that includes the task id.
-w <View>	No	View name. This might be needed when multiple views are in use and a single zone push is desired. If not specified, the value "Default" is used.
-z <Zone>	No, unless –w specified	Zone name. Set this option to generate a configuration for this selected zone, instead of the whole server.

Usage Examples

This example creates a task to push any changed zone files to the server `dns1`. The configuration file is pushed if needed. Also, by default, the configuration file and zone files are checked for errors.

```
$INCHOME/etc/cli/DnsConfigurationTask.sh –u joe –p joepwd –n dns1
```

This example creates a task to push any changed zone files to the server `dns1`. The configuration file is pushed if needed.

```
$INCHOME/etc/cli/DnsConfigurationTask.sh –u joe –p joepwd –c –n dns1
```

This example creates a task to push the zone file for zone `acme.com` to the server `dns1`. Also, by default, the zone file is checked for errors.

```
$INCHOME/etc/cli/DnsConfigurationTask.sh –u joe –p joepwd –n dns1 –z acme.com
```

This example creates a task to send all of resource records for the zone `acme.com` via Dynamic DNS to the server `dns1`.

```
$INCHOME/etc/cli/DnsConfigurationTask.sh –u joe –p joepwd –d –n dns1 –z acme.com
```

This example creates a task to send the changed resource records for the zone `acme.com` via Dynamic DNS to the server `dns1`.

```
$INCHOME/etc/cli/DnsConfigurationTask.sh –u joe –p joepwd –c –d –n dns1 –z acme.com
```

Return codes

If the task is scheduled successfully, no message is displayed to the console. Specify –v to see the task number in a displayed message. That task number can then be passed to the **`TaskStatus`** CLI to obtain the status of that task.

If the task is NOT scheduled successfully, a negative number is displayed in an error message printed to the console.

GlobalRollup

Overview

The **GlobalRollup** CLI allows the user to issue an immediate Global Rollup task to summarize collected utilization statistics and perform regression analysis.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.GlobalRollupCLI -u <userId> -p <pswd>
-t <#><period> [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/GlobalRollup.sh -u <userId> -p <pswd>
-t <#><period> [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/GlobalRollup.cmd -u <userId> -p <pswd>
-t <#><period> [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pswd>	Yes	Password
-?	No	Print help
-t <#><period>	Yes	Regression time periods. <#> indicates the number of periods. <period> indicates the period type, where acceptable values are D, W, M, Y (days, weeks, months, years).
-v	No	Produces verbose output, providing the task id.

Usage Examples

This example creates a task on the queue to perform a Global Rollup, and to use the last 90 days of data when forecasting future growth.

```
$INCHOME/etc/cli/GlobalRollup.sh -u joe -p joepwd -t 90d
```

This example creates a task on the queue to perform a Global Rollup, and to use the last 6 months of data when forecasting future growth.

```
$INCHOME/etc/cli/GlobalRollup.sh -u joe -p joepwd -t 6m
```

Return codes

If the task is scheduled successfully, an errorlevel of 0 is passed, along with a string including the task number. That task number can then be passed to the **TaskStatus** CLI to obtain the status of that task.

If the task is NOT scheduled successfully, the errorlevel returned will be a negative number, and an error message will be printed to the console.

GlobalSync

Overview

The **GlobalSync** CLI allows the user to issue an immediate Global Synchronization task for either network services or network elements.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.GlobalSyncCLI -u <userId> -p <pswd>
-t <type> [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/GlobalSync.sh -u <userId> -p <pswd>
-t <type> [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/GlobalSync.cmd -u <userId> -p <pswd>
-t <type> [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	Username
-p <pswd>	Yes	Password
-?	No	Print help
-t <type>	Yes	Indicates the type of Global Sync to perform. Acceptable values are **netelement** or **netservice**.
-v	No	Produces verbose output, providing the task id.

Usage Example

This example creates a task on the queue to perform a Global Synchronization of all network elements that are flagged in IPAM for inclusion in the Global Sync process.

```
$INCHOME/etc/cli/GlobalSync.sh -u joe -p joepwd -t netelement
```

This example creates a task on the queue to perform a Global Synchronization of all network services that are flagged in IPAM for inclusion in the Global Sync process.

```
$INCHOME/etc/cli/GlobalSync.sh -u joe -p joepwd -t netservice
```

Return codes

If the task is scheduled successfully, an errorlevel of 0 is passed, along with a string including the task number. That task number can then be passed to the **TaskStatus** CLI to obtain the status of that task.

If the task is NOT scheduled successfully, the errorlevel returned will be a negative number, and an error message will be printed to the console.

TaskStatus

Overview

The **TaskStatus** CLI allows the user to query the status of tasks.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.TaskStatusCLI -u <userId> -p <pswd>
-t <task number> [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/TaskStatus.sh -u <userId> -p <pswd>
-t <task number> [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/TaskStatus.cmd -u <userId> -p <pswd>
-t <task number> [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	Username
-p <pswd>	Yes	Password
-?	No	Print help
-t <task number>	Yes	The task number to query.
-v	No	Verbose output. Detailed information about the task will be displayed.

Usage Example

This example queries task `37` and returns a one-word string indicating the status:

```
INPROGRESS
```

```
$INCHOME/etc/cli/TaskStatus.sh -u joe -p joepwd -t 37
```

This example queries task `42` and returns detailed information about that task:

```
42/dhcp5/COMPLETE/2003-12-16 10:32:38.0/2003-12-16
10:35:38.0/00:03:00
```

```
$INCHOME/etc/cli/TaskStatus.sh -u joe -p joepwd -t 42 -v
```

Output

By default, **TaskStatus** returns the status of the queried task as:

- NOTSTARTED
- QUEUED
- INPROGRESS
- COMPLETE

- COMPLETEWITHERRORS
- ERROR

Using the '-v' parameter on the command line outputs detailed information about the task in the format:

```
<id> /<scope> /<status> /<starttime> /<completedtime> /<duration>
```

Return codes

`TaskStatus` also returns an errorlevel indicating the task status, corresponding to the following table:

Status	Errorlevel
NOTSTARTED	1
QUEUED	2
INPROGRESS	3
COMPLETE	4
COMPLETEWITHERRORS	5
ERROR	6

Updates

DetachBlock

Overview

The **DetachBlock** CLI enables the user to detach blocks from device containers. This CLI allows you to specify a file containing a list of blocks to detach.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH task number
com.diamondip.netcontrol.cli.DetachBlockCLI -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DetachBlock.sh -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DetachBlock.cmd -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	Username
-p <pswd>	Yes	Password
-?	No	Print help
-f <update filename>	Yes	The name of the CSV listing blocks to be detached. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-deleted) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.
-v <verbose>	No	Produce verbose output.

Usage Example

This example detaches blocks listed in the *blocks.csv* file, places into the *blocks.reject* file any records that could not be deleted, and report errors to the *detacherrors.txt* file.

```
$INCHOME/etc/cli/DetachBlock.sh -u joe -p joepwd -f blocks.csv -r blocks.reject
-e detacherrors.txt
```

File Format

Specify the name of the block, or its address and size, and the container it is to be detached from.

Col	Field	Accepted Values	Required
A	Block Name	The name of the block to be detached. This is typically the CIDR address: *Address space/Block size,* e.g. 10.1.2.0/24. However, if the block name was set manually during allocation, that value must be used instead.	Yes, unless block address and block size are specified
B	Block Address	The address space of the block to be detached.	Yes, unless block name is specified
C	Block Size	The size of the block in short-notation (e.g., 24 for a 255.255.255.0 network).	Yes, unless block name is specified
D	Container	The name of the container from which to detach the block. Names can be in either short or long format. Short format example: Dallas. Long format example: InControl/Texas/Dallas. Long format eliminates ambiguity in cases where there are duplicate container names.	Yes
E	Interface Address(es)	The specific address(es), for the interface IP address(es) to be detached. Multiple addresses can be specified by separating the values with the '|' character. For example, specify 2 interface addresses as 10.0.0.1|10.0.0.2.	No

DetachContainer

Overview

The **DetachContainer** CLI enables the user to detach device containers that are children in multiple points in the container tree from one of their parent containers. If the device container is only a child of one parent, then DeleteContainer must be used instead to delete the device container from the container tree entirely. This CLI allows you to specify a file containing a list of containers to detach.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH task number
com.diamondip.ipcontrol.cli.DetachContainerCLI -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DetachContainer.sh -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-v] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DetachContainer.cmd -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	Username
-p <pswd>	Yes	Password
-?	No	Print help
-f <update filename>	Yes	The name of the CSV listing containers to be detached. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected (non-detached) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.
-v <verbose>	No	Produce verbose output.

Usage Example

This example detaches containers listed in the *containers.csv* file, places into the *containers.reject* file any records that could not be detached, and report errors to the *detacherrors.txt* file.

```
$INCHOME/etc/cli/DetachContainer.sh -u joe -p joepwd -f containers.csv -r containers.reject
-e detacherrors.txt
```

File Format

Specify the full path of the container to be detached, any V4 DHCP servers, and any V6 DHCP servers to use in case pools or subnets are invalidated due to the change in container parenting.

Col	Field	Accepted Values	Required
A	Full Path of the Container Name	The full path name of the device container to be detached. The full path is required here as that determines which parent to detach the container from. The container must have more than one parent.	Yes
B	V4 DHCP Server Name	The V4 DHCP server to use for any pools or subnets whose DHCP server has been invalidated by the change in parent hierarchy. If this is not specified, no changes will be made to the assigned DHCP servers.	No
C	V6 DHCP Server Name	The V6 DHCP server to use for any pools or subnets whose DHCP server has been invalidated by the change in parent hierarchy. If this is not specified, no changes will be made to the assigned DHCP servers.	No

JoinBlock

Overview

The **`JoinBlock`** CLI allows the user to join existing, adjacent blocks, forming a larger block. The blocks must be in the same container, and of the same type. This CLI allows you to specify a single block on the command line.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.JoinBlockCLI -u <userId> -p <pswd>
-b <block Name> [-c <container name>] [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/JoinBlock.sh -u <userId> -p <pswd>
-b <block Name> [-c <container name>] [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/JoinBlock.cmd -u <userId> -p <pswd>
-b <block Name> [-c <container name>] [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	Username
-p <pswd>	Yes	Password
-?	No	Print help
-r <rejects file>	No	The name of the file that rejected (non-deleted) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.
-b <block name>	Yes	The name of the first block to be joined. This is typically the CIDR address, e.g. 10.1.2.0/24. However, if the block name was set manually during allocation, that value must be used instead.
-c <container name>	Must be specified if the block name is not unique.	The name of the container holding the block. If this parameter is supplied, the container is searched for the block to join. Otherwise, the whole system is searched.

Usage Example

This example joins the block `10.1.2.0/24` to the next adjacent block in the same container. As mentioned above, this example assumes that there is only one block with this name in the system.

```
$INCHOME/etc/cli/JoinBlock.sh -u joe -p joepwd -b 10.1.2.0/24
```

ModifyAddrpool

Overview

The **ModifyAddrpool** CLI alters existing address pools in the system.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.ModifyAddrpoolCLI -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ModifyAddrpool.sh -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ModifyAddrpool.cmd -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	Username
-p <pswd>	Yes	Password
-?	No	Print help
-f <update filename>	Yes	The name of the CSV listing addrpools to be modified. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

```
$INCHOME/etc/cli/ModifyAddrpool.sh -u joe -p joepwd -f updateaddrpools.txt
-r updateaddrpools.reject -e updateaddrpool.err
```

Output

If successful, the CLI updates the address pools per the input file and exits.

File Format

The **ModifyAddrpool** CLI uses attribute-value pairs to specify the records and fields to be changed. Each line in the input file must have a set of attribute-value pairs to locate the address pool to be changed and a second set specifies what fields to change and their new values.

The following table lists the available attributes for address pools and also indicates which can be used to locate a record.

Field	Attribute Name	Accepted Values	Locator Field
Start Address	startAddr	The IP Address of the first address in the pool. This address must be in a block with an In-Use/Deployed status.	Required
End Address	endAddr	The IP Address of the last address in the pool. This address must be in the same block as the Start Address. In addition, the Start and End addresses must not overlap any other pools.	No
Prefix Length	prefixLength	The CIDR size of an IPv6 address pool. When the end of the pool is calculated, the end address must be in the same block as the start address. Also, the pool that would result must not overlap any other pools.	No
Address Pool Type	type	One of "Dynamic DHCP", "Automatic DHCP", "Static", "Reserved", "Dynamic NA DHCPv6", "Automatic NA DHCPv6", "Dynamic TA DHCPv6", "Automatic TA DHCPv6"	No
Name	name	Address Pool name. Defaults to "Start Address-End Address"	No
~~Share Name~~ *Deprecated*	~~sharename~~	~~The name used to link address pools together.~~	No
Container	container	The name of the container that holds the block in which the pool is defined. This is required only if there is overlapping address space in use and the start address is in overlapping space. The container is then used to uniquely determine the block that will contain the address pool.	Yes, but not required unless Start address is not unique.
Primary Net Service	primaryNetService	The name of the DHCP server that will serve addresses from this pool	No
Failover Net Service	failoverNetService	The name of the failover DHCP server that will serve addresses from this pool	No
DHCP Option Set	dhcpOptionSet	The name of an Option Set used with this pool.	No
DHCP Policy Set	dhcpPolicySet	The name of a Policy Set used with this pool.	No

Allow DHCP Client Classes	allowClientClasses	A list of Client Classes that are allowed in this address pools. Separate the list entries with a vertical bar "|".	No
Deny DHCP Client Classes	denyClientClasses	A list of Client Classes that are NOT allowed in this address pools. Separate the list entries with a vertical bar "|".	No
Overlap Interface IP	overlapInterfaceIp	Flag to allow the DHCPv6 pool to overlap an interface address. This flag may be set only if pool is managed by a CNR DHCPv6 server. This is ignored for DHCPv4 pools.	No

Each input line specifies the locator attribute-value pairs, followed by a colon, followed by the modification attribute-value pairs. For example, to change the ending address for the pool starting at address `10.1.2.3`:

```
startAddr=10.1.2.3:endAddr=10.1.2.15
```

Separate multiple attribute-value pairs with commas. For example, to change both the end address and the DHCP Server for the address pool starting at `10.1.2.3`:

```
startAddr=10.1.2.3:endAddr=10.1.2.15,primaryNetService=newserver
```

This applies to the locator fields as well. For example, to change the end address for the pool starting at `192.168.0.2` in Container `Private`:

```
startAddr=192.168.0.2,container=Private:endAddr=192.168.0.31
```

Some values are lists. Separate the list elements with a vertical bar. For example, to allow two Client Classes for the pool starting at `10.1.2.3`:

```
startAddr=10.1.2.3:allowClientClasses=allow1|allow2
```

For fields that are lists, existing values, if any, may be replaced or merged. For example, for the pool starting at `10.1.2.3`, to replace `allow1` and `allow2` with `allow3` in the example above write:

```
startAddr=10.1.2.3:allowClientClasses=allow3
```

To update only some of the values in a list use the notation += when specifying the attribute and value. In the example above to allow a client class `allow4` while keeping the `allow3`, write:

```
startAddr=10.1.2.3: allowClientClasses+=allow4
```

To remove a value, specify the attribute but leave the value empty. For example, to remove the `allowClientClasses` from the above pool:

```
startAddr=10.1.2.3:allowClientClasses=
```

V6 Addresses

To specify an IPV6 address, # must be used as the separator between the locator pairs and the modification pairs, since : is part of the address:

```
startAddr=2001::1:0#primaryNetService=newserver
```

ModifyBlock

Overview

The **ModifyBlock** CLI alters existing blocks in the system.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.netcontrol.cli.ModifyBlockCLI -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ModifyBlock.sh -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ModifyBlock.cmd -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	Username
-p <pswd>	Yes	Password
-?	No	Print help
-f <update filename>	Yes	The name of the CSV file describing the updates. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

```
$INCHOME/etc/cli/ModifyBlock.sh -u joe -p joepwd -f updateblocks.txt
-r updateblocks.reject -e updateblock.err
```

Output

If successful, the CLI updates the blocks per the input file and exits.

File Format

The **ModifyBlock** CLI uses attribute-value pairs to specify the records and fields to be changed. Each line in the input file must have a set of attribute-value pairs to locate the block to be changed, and a second set specifies what fields to change and their new values.

The following table lists the available attributes for blocks, and also indicates which can be used to locate a record.

Field	Attribute Name	Accepted Values	Locator Field/ Modify?
Block Address	blockAddr	The starting address of the block	Yes / No

ModifyBlock

Field	Attribute Name	Accepted Values	Locator Field/ Modify?
Block Name	blockName	The Name of the block	Yes / Yes
Block Size	blockSize	The CIDR size of the block, for example, "24"	Yes / No
Block Status	blockStatus	Block Status, one of "Free", "Reserved", "Aggregate", "Deployed", "FullyAssigned"	Yes / Yes
Block Type	blocktype	A valid block type name. (Validation associated with modifying a block's block type will be applied.)	No / Yes
Container	container	An array of container names	Yes/ Yes
ignoreErrors	ignoreErrors	True/false. If true, any errors occurring during a block reparent related to inconsistent user defined field templates will be ignored. For instance, moving a block with a container/blocktype information template to a container which does not have the same template will normally cause an error. To ignore the error (and lose the user defined fields associated with the old container) set the flag to ignoreErrors=true.	No/No
Interface Name	interfaceName	The target interface name during a reparent of a Device Container.	Yes / Yes
Description	description	The block description. Use "\n" to separate lines.	No/ Yes
Exclude from Discovery	excludeFromDiscovery	Flag indicating if this subnet should be included in Host Discovery tasks. Accepted values are **true** or **false**. **Valid only for Deployed blocks.**	No/ Yes
Organization Id	organizationId	Org ID of the RIR Organization	No/ Yes
RIR	rir	Name of the RIR Organization	No/ Yes
Root Block Type	rootBlocktype	The Block Type for the block.	No/ Yes
SWIP Name	swipname	SWIP Name for ARIN blocks	No/ Yes
User Defined Fields	userDefinedFields	Array of user defined fields. Each element in the array has the format "name=value" where "name is the UDF tag name.	No /Yes
Subnet	subnet	Block subnet policies. See below for syntax. **Valid only for Deployed blocks.**	No/Yes
Primary Subnet Flag	primarySubnet	Flag indicating if this subnet is primary. Accepted values are true or false. **Valid only for Deployed blocks.**	No/Yes
Allocation Template Name	allocationTemplate Name	For a block with blockStatus=**Deployed**, or if changing the blockStatus to **Deployed**, the name of the allocation template to use to create address pools from the newly created block.	No/Yes

Field	Attribute Name	Accepted Values	Locator Field/ Modify?
Address Allocation Details	addrDetails	Define attributes of any address allocations of the specified allocation template. See below for syntax.	No/Yes
NonBroadcast Flag	nonBroadcast	Flag indicating if this is a Non-Broadcast subnet. Non-broadcast subnets are allowed to assign devices to the subnet and broadcast addresses. Accepted values are **true** or **false**. If not specified, defaults to **false.** **Valid for IPv4 Deployed blocks only.**	No/Yes

Each input line specifies the locator attribute-value pairs, followed by a : or # , followed by the modification attribute-value pairs. For example, to change the description for the block starting at address `10.1.2.3`:

```
blockAddr=10.1.2.3:description="updated description"
```

Separate multiple attribute-value pairs with commas. For example, to change both the description and the Organization Id for the block starting at `10.1.2.3`:

```
blockAddr=10.1.2.3:description="updated description", rir=nationalISP
```

This applies to the locator fields as well. For example, to change the block status for a block:

```
blockAddr=10.1.2.3:blockStatus=Reserved
```

Some values are lists. Separate the list elements with a vertical bar. For example, to modify two user defined fields for the block at `10.1.2.3`:

```
blockAddr =10.1.2.3:userDefinedFields="state=PA"|"city=Exton"
```

For values that are lists, existing values, if any, may be replaced or merged. For example, for the block at `10.1.2.3` to replace theentire contents of the existing userDefinedFields write:

```
blockAddr =10.1.2.3:userDefinedFields="state=PA"|"city=Exton"
```

To update only some of the values in a list use the notation += when specifying the attribute and value. In the example above to update the city as `Devon` without changing or removing the state value, write:

```
blockAddr =10.1.2.3:userDefinedFields+="city=Devon"
```

To remove a value, specify the attribute but leave the value empty. For example, to remove the description from the above block:

```
blockAddr =10.1.2.3: description =
```

V6 Addresses

To modify an IPV6 block, `#` must be used as the separator between the locator pairs and the modification pairs, since `:` is part of the address:

```
blockAddr=3FFE:0000:0000:0010::#description="updated V6 description"
```

Updating Interface Addresses

To modify a block's interface address(es), use ImportChildBlock with the overwrite option.

Reparenting a Block via ModifyBlockCLI

The **ModifyBlock** CLI may also be used to reparent a block by specifying a target container name different than the current parent container name. In the case of reparenting blocks contained in device containers, a target interface name must also be provided.

For example, to reparent the block starting at `192.168.192.0` to the new container `MovedTo`:

```
blockAddr=192.168.192.0:container=MovedTo
```

To reparent the block starting at `192.168.196.134` from a device container to the new device container `Router1` on interface `routerInterface1`:

```
blockAddr=192.168.196.134:container=Router1,interfaceName=routerInterface1
```

Note: The Modify Block will either reparent, when a new container name different than current parent container has been specified, **or** modify the block without a reparent. Both cannot be performed during the same modify block invocation.

Modifying Block Subnet Policies

Use the subnet field to specify changes to block subnet policies. Subnet uses a nested data structure syntax. The attributes of subnet policies are surrounded by braces.

For example, to update the policies for a block starting at `192.168.196.134` (line wrapped for readability):

```
blockAddr=192.168.192.0: description="modifying policies",subnet={DHCPOptionsSet=Global
Option Set,DHCPPolicySet=Standard ISC DHCP 3.0 Policy
Set,DNSServers=ServerABC,defaultGateway=192.80.0.1,failoverDHCPServer=DHCPFailoverServer,fo
rwardDomains=test.com|example.com,forwardDomainTypes=Default|Internal,primaryDHCPServer=DHC
PServer,false,primaryWINSServer=192.80.0.2,reverseDomains=10.in-addr.arpa.|10.in-
addr.arpa.,reverseDomainTypes=Default|Internal}
```

The attributes for subnet are:

Field	Attribute Name	Accepted Values	Locator Field
DHCP Option Set	DHCPOptionsSet	The name of a DHCP Option Set defined within IPAM that should apply to this subnet.	No
DHCP Policy Set	DHCPPolicySet	The name of a DHCP Policy Set defined within IPAM that should apply to this subnet.	No
DNS Servers	DNSServers	The list of default DNS Servers for this subnet. This list is supplied to DHCP clients on this subnet. The server name or IP Address is valid. For multiple servers, separate the server names with a vertical bar (|).	No

Field	Attribute Name	Accepted Values	Locator Field
Default Gateway	defaultGateway	The default gateway address for this subnet. This address is supplied to DHCP clients on this subnet. This can be a comma-separated list of IP Addresses, surrounded by double-quotes.	No
Failover DHCP Server	failoverDHCPServer	The name or IP Address of the failover DHCP Server for this address space.	No
Forward Domain Types	forwardDomainTypes	The list of forward domain types corresponding with the list in the forwardDomains field, separated by a vertical bar (|). Not required when using the default domain type.	No
Forward Domain Names	forwardDomains	The list of DNS forward domains for this address space, separated by a vertical bar (|). Use forwardDomainTypes to specify the corresponding domain types.	No
Primary DHCP Server	primaryDHCPServer	The name or IP Address of the primary DHCP server for this address space.	No
Cascade Primary DHCP Server	cascadePrimaryDHCPServer	If address pool or individual IP address objects within the subnet reference a specific DHCP server, this attribute can be used to allow the 'primaryDHCPServer' attribute value to "cascade" to these address pool and IP address objects. Specify **true** to apply this update. Note that address pool and IP address objects that are configured with "Same as Subnet" for the primary DHCP server will be unaffected.	No
Primary WINS Server	primaryWINSServer	The IP Address of the primary WINS server for this subnet. Used to provide this information to DHCP for Dynamic Address types. Multiple WINS servers may be specified, separated by commas.	No
Reverse Domain Types	reverseDomainTypes	The list of reverse domain types corresponding with the list in the reverseDomains field, separated by a vertical bar (|). Not required when using the default domain type.	No

Field	Attribute Name	Accepted Values	Locator Field
Reverse Domain Names	reverseDomains	The list of DNS reverse domains for this address space, separated by a vertical bar (|). Use reverseDomainTypes to specify the corresponding domain types.	No

Applying an Allocation Template

Use the **addrDetails** field to optionally specify attributes of any address allocations within the allocation template specified by **allocationTemplateName**. You can apply a template and specify address details when changing a block's status to Deployed, or when modifying a block that is already of status Deployed. The **addrDetails** uses a nested data structure syntax, so its attributes are surrounded by braces.

For example (line wrapped for readability):

```
blockAddr=192.168.192.0: blockStatus=Deployed, description="applying allocation template",
allocationTemplateName=Standard DHCP,
addrDetails={startingOffset=3,offsetFromBeginningOfSubnet=true:netserviceName=DHCPServer}
```

The attributes for subnet are:

Field	Attribute Name	Accepted Values	Locator Field
Starting Offset	startingOffset	Identify the address allocation within the template. This must match the specification in the Allocation Template.	Yes
Is the starting offset from the beginning of subnet?	offsetFromBeginningOf Subnet	Specify **true** or **false**. This must match the specification in the Allocation Template.	Yes
Network Service Name	netserviceName	The name of the network service for this address allocation.	No

ModifyContainer

Overview

The **ModifyContainer** CLI alters existing containers in the system.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.ModifyContainerCLI -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ModifyContainer.sh -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ModifyContainer.cmd -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	Username
-p <pswd>	Yes	Password
-?	No	Print help
-f <update filename>	Yes	The name of the CSV file describing the modifications. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.
-v	No	Produces verbose output.

Usage Example

```
$INCHOME/etc/cli/ModifyContainer.sh -u joe -p joepwd -f updatecontainers.txt
-r updatecontainers.reject -e updatecontainers.err
```

Output

If successful, the CLI updates the containers per the input file and exits.

File Format

The **ModifyContainer** CLI uses attribute-value pairs to specify the records and fields to be changed. Each line in the input file uses an attribute-value pair to locate the container to be changed and a set of attribute-value pairs that specifies which attributes to modify.

If an attribute is not included in the modifier set, then its value is not changed.

The following table lists the available attributes for containers and also indicates which can be used to locate a record.

ModifyContainer

Field	Attribute Name	Accepted Values	Locator Field
Container Name	containerName	The name of the container. Names can be in either short or long format. Short format example: **Dallas**. Long format example: **InControl/Texas/Dallas**. Long format eliminates ambiguity in cases where there are duplicate container names. If using the long format, the name must be the complete path beginning at the top of the container tree.	Yes
Container Description	description	A brief description of the container. Use "\n" to separate lines.	No
Parent Container	parentName	The name of the parent container for this container. Names can be in either short or long format. Modification indicates the container should be moved to a different parent container.	No
Information Template	informationTemplate	A list of a pre-existing information templates to be associated with this container, separated by '|'.	No
Allowed Block Types	allowedBlockTypes	A list of the valid block types for this container, separated by '|'. To specify information templates for block types, use the blockTypeInfoTemplates field.	No
Block Type Information Templates	blockTypeInfo Templates	A list of the information templates to be used for block types, separated by '|'. Specify the templates in the same order as the block types in the allowedBlockTypes field. If no template is to be associated with a particular block type, leave it blank, but include the separator. For example: **blockTypeInfoTemplates= |template2** In this example, the first block type listed in allowedBlockTypes does not use an information template.	No
Allowed Root Block Types	allowedRootBlock Types	A list of the block types enabled for root block creation, separated by '|'.	No

Field	Attribute Name	Accepted Values	Locator Field
Allowed Block Types from Parent	allowedAllocFrom ParentBlocktypes	A list of the block types that can be used for space allocation from the parent container, separated by '|'.	No
Require SWIP Name Block Types	requireSWIPName BlockTypes	A list of the block types for which SWIP names are required, separated by '|'.	No
Allowed Device Types	allowedDeviceTypes	A list of the valid device types for this container, separated by '|'. To specify information templates for devices, use the deviceInfoTemplates field.	No
Device Information Templates	deviceInfoTemplates	A list of the information templates to be used for devices, separated by '|'. Specify the templates in the same order as the device types in the allowedDeviceTypes field. If no template is to be associated with a particular device type, leave it blank, but include the separator. For example: **deviceInfoTemplates= |template2** In this example, the first block type listed in allowedBlockTypes does not use an information template.	No
User Defined Fields	userDefinedFields	List of name=value parameters, separated by '|'. If the UDF type is Checkbox, the valid values are **on** and **off**. If the UDF type is Textarea, use "\n" to separate lines.	No
Ignore disallowing a block type in use	ignoreBlocktypeInUse	Set this to "true" when disallowing a block type in use by the container, indicated by the list in the allowedBlockTypes field.	No
Maintain History Records	maintainHistoryRecs	Specify whether or not Container History and Block History records will be kept for all appropriate block types. The history records are created each time the Global Utilization Rollup task is run. Accepted values are **true** or **false**. If not specified, defaults to **false**.	No

Field	Attribute Name	Accepted Values	Locator Field
Replace DHCP Servers	replaceDhcpServer	This attribute is used only when utilizing the IPAM feature of attaching DHCP servers to Containers, and when the modify operation involves moving a container. In this situation, Subnet, Address Pool, and IP Address objects that are moved as a result of the container move may become "invalid" with respect to the DHCP servers attached to the target container or target container's nearest anscestor. In order to update these objects' associated DHCP servers as part of the container move operation, specify the name of a DHCP server which is "valid" for the target container. This will update only dynamic V4 objects and must be a V4 capable DHCP server.	No
Replace V6 DHCP Servers	replaceDhcpServerV6	This is similar to the above "replaceDHCPServer" field, but will update any dynamic V6 objects and must be a V6 capable DHCP server.	No
Apply Replace DHCP to Multiparented Device Containers	applyDhcpToMultiparentDevContainer	This attribute is used only in conjunction with the 'replaceDhcpServer' attribute, when replacing DHCP servers in objects during a container move operation. This flag indicates if those changes should affect objects attached to multiparented device containers, which may have differing DHCP server associations along each container anscestory. Note: Multiparented device containers cannot be moved using CLIs.This attribute applies when moving a logical container with a mulitparented device container along the logical container ancestory.	No

Each input line specifies the locator attribute-value pair, followed by a colon, followed by the modification attribute-value pairs. For example, to change the description for the container `East`:

```
containerName=East:description=new description
```

Separate multiple attribute-value pairs with commas. For example, to change both the description and the information template for the container `East`:

```
containerName=East:description=new description,informationTemplate=templateName
```

For fields that are lists, separate the list elements with a vertical bar. For example, to set block types for the container `East`:

```
containerName=East:allowedBlockTypes=Any|blocktyp1|blocktyp2
```

User Defined Fields use a nested `attribute=value` syntax. For example, to set user defined fields for container `East`:

```
containerName=East:userDefinedFields=”udf1=value1”|”udf2=value2”
```

For fields that are lists, existing values, if any, may be replaced or merged. For example, for container `East`, to replace the entire contents of the existing user defined fields, write:

```
containerName=East:userDefinedFields=”state=PA”|”city=Exton”
```

To update only some of the values in a list use the notation += when specifying the attribute and value. In the example above, to update the city as `Devon` without changing or removing the state value, write:

```
containerName=East:userDefinedFields+=”city=Devon”
```

To remove a value, specify the attribute but leave the value empty. For example, to remove the description for container `East`:

```
containerName=East:description=
```

To modify information templates for block and device types, specify the list of templates in the same order as the list of block or device types. For example:

```
containerName=East:allowedBlockTypes=Any|blocktype1|blocktype2,blockTypeInfoTemplates=|newT
emplate|newTemplate,allowedDeviceTypes=Printer|Router|Switch,deviceInfoTemplates=xTemplate|
xtemplate|newTemplate
```

In the above example, block type `Any` has no template, while `blocktype1` and `blocktype2` use `newTemplate`. Device types `Printer` and `Router` use `xTemplate`, and `Switch` uses `newTemplate`.

ModifyDevice

Overview

The **ModifyDevice** CLI alters existing devices in the system.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH

com.diamondip.netcontrol.cli.ModifyDeviceCLI -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ModifyDevice.sh -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ModifyDevice.cmd -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	Username
-p <pswd>	Yes	Password
-?	No	Print help
-f <update filename>	Yes	The name of the CSV file describing the modifications. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

```
$INCHOME/etc/cli/ModifyDevice.sh -u joe -p joepwd -f updatedevices.txt
-r updatedevices.reject -e updatedevices.err
```

Output

If successful, the CLI updates the devices per the input file and exits.

File Format

The **ModifyDevice** CLI uses attribute-value pairs to specify the records and fields to be changed. Each line in the input file must have a set of attribute-value pairs to locate the device to be changed and a second set that specifies what attributes to modify.

If an attribute is not included in the modifier set, then its value is not changed.

The following table lists the available attributes for devices and also indicates which can be used to locate a record. If you want to modify an attribute not available in this CLI, use the ImportDevice CLI with the overwrite feature.

Field	Attribute Name	Accepted Values	Locator Field
IP Address	ipAddress	The IP Address of the Device.	Yes
MAC Address	MACAddress	The Hardware MAC Address of the device.	Yes
Address Type	addressType	The address type of this device. Accepted values are: **Static**, **Dynamic DHCP**, **Automatic DHCP**, **Manual DHCP**, and **Reserved.**	No
Container	container	The Container that holds the block for the IP Address.	Yes, if IP Address in overlapping space.
Description	description	Text Description of the device. Use "\n" to separate lines.	No
Device Type	deviceType	The device type of the device. Should be one of the values defined in the system. Defaults to "Unspecified"	No
Domain Name	domainName	The name of the domain for this device.	No
Domain Type	domainType	The domain type of the domain. Defaults to "Default"	No
Duplicate Warning	dupWarning	Set this to "true" if duplicate warnings should be ignored.	No
Host Name	hostname	The device Host name.	Yes
Hardware Type	hwType	Ethernet or Token Ring	No
Resource Record Flag	resourceRecordFlag	Accepted values are **true** or **false**. If not specified, defaults to **false**. The resource records associated with the device will be updated when the hostname or IP Address changes regardless of this setting. If the flag is "true" and there are no resource records associated with the device, resource records will be generated for the device. Note that the domain name must be specified if the block policy has no forward domains. Also, the reverse domain must exist in order for the PTR record to be added.	No
User Defined Fields	userDefinedFields	List of name=value parameters, separated by a vertical bar. If the UDF type is Checkbox, the valid values are **on** and **off**. If the UDF type is Textarea, use "\n" to separate lines.	No

Field	Attribute Name	Accepted Values	Locator Field
Aliases	aliases	List of alias names, separated by a vertical bar. Only used if the resource record flag is set.	No
Interfaces	interfaces	List of interfaces. See below for syntax. Use this to modify a multi-home device.	No

Each input line specifies the locator attribute-value pairs, followed by a colon, followed by the modification attribute-value pairs. For example, to change the hostname for the device at address `10.1.2.3`:

```
ipAddress=10.1.2.3:hostname=newhostname
```

Separate multiple attribute-value pairs with commas. For example, to change both the hostname and the description for the device at `10.1.2.3`:

```
ipAddress=10.1.2.3:hostname=newhostname,description="New Host"
```

This applies to the locator fields as well. For example, to change the hostname for the device at `192.168.0.2` in Container `Private`:

```
ipAddress=192.168.0.2,container=Private:hostname=newhostname
```

Aliases, user defined fields, and interfaces values are lists. Separate the list elements with a vertical bar. For example, to set two aliases for the device at `10.1.2.3`:

```
ipAddress=10.1.2.3:aliases=alias1|alias2
```

User Defined Fields use a nested `attribute=value` syntax. For example, to set a user defined field for IP Address `10.1.2.3`:

```
ipAddress=10.1.2.3:userDefinedFields="udf1=value1"|"udf2=value2"
```

For fields that are lists, existing values, if any, may be replaced or merged. For example, for IP Address `10.1.2.3`, to replace entire contents of the existing user defined fields, write:

```
ipAddress=10.1.2.3:userDefinedFields="state=PA"|"city=Exton"
```

To update only some of the values in a list use the notation += when specifying the attribute and value. Please note, the += notation does not apply to Interfaces when modifying multi-home devices. In the example above, to update the city as `Devon` without changing or removing the state value, write:

```
ipAddress=10.1.2.3:userDefinedFields+="city=Devon"
```

To remove a value, specify the attribute but leave the value empty. For example, to remove the description from the above device:

```
ipAddress=10.1.2.3:description=
```

Modifying a Multi-Home Device

Working with Multi-Home devices is more complex. To locate a multi-home device, specify either the device's host name, or *any* of its IP Addresses or MAC addresses.

To update the IP Addresses or MAC Addresses, do NOT use the ipAddress or MACAddress primary attributes. Instead, specify them as attributes of the interfaces.

Interfaces use a nested data structure syntax. For example:

To update the interfaces for the device with hostname newhostname and to 'modify' their IP Address, use colons between the interface attributes (line wrapped for readability):

```
hostname=newhostname:interfaces={name=Default}|{name=eth0:ipAddress=10.1.2.3}|
{name=eth1:ipAddress=10.1.2.4}
```

To update the interfaces for the device with hostname newhostname and to 'add' new interfaces, use commas between the interface attributes (line wrapped for readability):

```
hostname=newhostname:interfaces={name=Default}{name=eth0,ipAddress=10.1.2.3}|
{name=eth1,ipAddress=10.1.2.4}
```

The attributes for interfaces are:

Field	Attribute Name	Accepted Values	Locator Field
Name	Name	Interface Name	Yes
IP Address	ipaddress	IP Address of the interface	Yes
MAC Address	macAddress	Hardware MAC Address of the interface	Yes
Hardware Type	hwType	Ethernet or Token Ring	No
Sequence	Sequence	Reserved	No

Note: It is possible to convert a single-homed device into a multi-homed device by specifying the interfaces as shown above. To do this, use the device's hostname or IP Address as a locator, and then specify the new interfaces along with their attributes as given in the table above. It is a must, to include the **{name=Default}** interface in the syntax, such as:

```
hostname=newhostname:interfaces={name=Default}|{name=eth0,ipAddress=10.1.2.3}|
{name=eth1,ipAddress=10.1.2.4}
```

Note: When adding or updating interfaces, all the existing interfaces need to be listed. Any interfaces not specified will get deleted.

V6 Addresses

To specify an IPV6 address, # must be used as the separator between the locator pairs and the modification pairs, since : is part of the address:

```
ipAddress= 2001::2#hostname=newhostname,description="New Host"
```

ModifyDeviceResourceRecord

Overview

The **ModifyDeviceResourceRecord** CLI alters existing device resource records in the system.

Usage

Direct

```
$INCHOME/jre/bin/java –cp $CLASSPATH
com.diamondip.ipcontrol.cli.ModifyDeviceResourceRecord –u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ModifyDeviceResourceRecord.sh –u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ModifyDeviceResourceRecord.cmd –u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	Username
-p <pswd>	Yes	Password
-?	No	Print help
-f <update filename>	Yes	The name of the CSV file describing the modifications. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

```
$INCHOME/etc/cli/ModifyDeviceResourceRecord.sh –u joe –p joepwd
-f updateeviceresourcerecs.txt -r updatedeviceresourcerecs.reject
–e updatedeviceresourcerecs.err
```

Output

If successful, the CLI updates the device resource records per the input file and exits.

File Format

The **ModifyDeviceResourceRecord** CLI uses attribute-value pairs to specify the records and fields to be changed Each line in the input file must have a set of attribute-value pairs to locate the resource record to be changed and a second set specifies what fields to change and their new values.

The following table lists the available attributes for device resource records and also indicates which can be used to locate a record.

Field	Attribute Name	Accepted Values	Locator Field/ Required?
Domain Name	domain	The name of the domain for this resource record.	Yes/Yes
Domain Type	domainType	The domain type of the domain. Defaults to "Default"	Yes/No
Owner	owner	The OWNER section of the resource record. Note that this section is specific to the type of resource record. Refer to the appropriate RFC for exact text that should be entered.	Yes/Yes
Host Name	hostname	The device host name.	Yes/Yes, unless ipAddress is specified
IP Address	ipAddress	The IP Address of the Device.	Yes/Yes, unless hostname is specified
Container	container	The name of the container that holds the device. This is required only if there is overlapping address space in use and the ip address is in overlapping space. The container is then used to uniquely determine the device.	Yes/No
TTL	TTL	The Time to Live	No/No
Class	resourceRecClass	The value currently supported is IN. If not specified, defaults to IN.	Yes/No
Resource Record Type	resourceRecType	The type of resource record being updated.	Yes/Yes
Data	data	The text for the DATA area of the resource record. Note that this section is specific to the type of resource record. Refer to the appropriate RFC for exact text that should be entered.	Yes/No, unless required to uniquely identify the record.
Comment	comment	Text to be appended to the resource record.	No/No

Each input line specifies the locator attribute-value pairs, followed by a colon, followed by the modification attribute-value pairs. For example, to change the data and TTL for a record:

```
domain=40.10.in-
addr.arpa.,domainType=Default,owner=10.10,hostname=router00001,resourceRecClass=IN,resource
RecType=A:data="newDNS.ins.com",TTL=2400
```

This applies to the locator fields as well. For example, to change the owner for a record:

```
domain=40.10.in-
addr.arpa.,domainType=Default,owner=10.10,hostname=router00001,resourceRecClass=IN,resource
RecType=A:owner=30.10
```

To remove a value, specify the attribute but leave the value empty. For example, to remove the description from the above device:

```
domain=40.10.in-
addr.arpa.,domainType=Default,owner=10.10,hostname=router00001,resourceRecClass=IN,resource
RecType=A:comment=
```

V6 Addresses

To specify an IPV6 address, # must be used as the separator between the locator pairs and the modification pairs, since : is part of the address:

```
domain=example.com,domainType=Default,owner=router00015,ipAddress=3FFE:0000:0000:0015::,res
ourceRecClass=IN,resourceRecType=AAAA#TTL=2400
```

Note on overlapping space:

If the device is in overlapping space, and the device in both spaces have A records with identical owners, if the administrator's role does not indicate "Ignore" for "Allow Duplicate A Record (Owner) Checking", this CLI will fail with "Duplicate Resource Record".

ModifyDhcpServer

Overview

The **ModifyDhcpServer** CLI alters existing DHCP servers in the system.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.ModifyDhcpServerCLI -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ModifyDhcpServer.sh -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ModifyDhcpServer.cmd -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	Username
-p <pswd>	Yes	Password
-?	No	Print help
-f <update filename>	Yes	The name of the CSV file describing the modifications. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

```
$INCHOME/etc/cli/ModifyDhcpServer.sh -u joe -p joepwd -f updatedhcp.txt
-r updatedhcp.reject -e updatedhcp.err
```

Output

If successful, the CLI updates DHCP servers per the input file and exits.

File Format

The **ModifyDhcpServer** CLI uses attribute-value pairs to specify the records and fields to be changed. Each line in the input file must have a set of attribute-value pairs to locate the DHCP server to be changed and a second set specifies what attributes to change.

If an attribute is not included in the modifier set, then its value is not changed.

The following table lists the available attributes for DHCP servers and also indicates which can be used to locate a record.

ModifyDhcpServer

Field	Attribute Name	Accepted Values	Locator Field
Name	name	The Name of the DHCP server	Yes
IP Address	ipAddress	The IP Address of the DHCP Server.	Yes
Product	product	The DHCP server product name defined in IPAM.	No
Agent	agent	The name of an agent defined in IPAM.	No
Default Threshold	defaultThreshold	0-100	No
Global Sync	globalSync	True or False	No
Configuration Path	configPath	Valid file name on host system.	No
Lease Path	leasePath	Valid file name on host system.	No
Start Script	startScript	Valid file name on host system.	No
Stop Script	stopScript	Valid file name on host system.	No
Collection Type	collectionType	SCP or FTP or CNRSDK	No
Collection Port	collectionPort	1-65535	No
Collection User	collectionUser	Valid account for SCP/FTP on Executive.	No
Collection Password	collectionPassword	Password for collection user.	No
CLI Command	cliCommand	Valid file name on host system.	No
CLI User Name	cliUserName	Valid user for collection program.	No
CLI Password	cliPassword	Password for collection program	No
CLI Arguments	cliArgs	Arguments to pass to collection command.	No
Dynamic DNS	ddns	Specify 'interim', 'standard', or 'lastin' to enable dynamic DNS updates when this server issues a lease. Defaults to 'none'.	No
DHCP Option Set	optionSet	Valid Option Set defined in IPAM	No
DHCP Policy Set	policySet	Valid Policy Set defined in IPAM.	No
DHCP Client Classes	clientClasses	Valid DHCP Client Classes defined in IPAM, separated by a vertical bar ("|").	No
DHCP Failover IP Address	failoverIpAddress	Valid IP Address	No
DHCP Failover Port	failoverPort	1-65535	No
Configuration Pre-Extension	beginExtension	Text or file name. File names must be prefixed by "file:".	No
Configuration Post-extension	endExtension	Test or file name. File names must be prefixed by "file:".	No
V4V6Both	v4v6both	'v4', 'v6' or 'both'	No

Each input line specifies the locator attribute-value pairs, followed by a colon, followed by the modification attribute-value pairs. To leave an attribute unchanged, simply omit it from the modification attribute-value pairs. For example, to change the Client Classes for the server at address `10.1.2.3`:

```
ipAddress=10.1.2.3:clientClasses=allow1|allow2|deny3
```

For fields that are lists separated by vertical bars, existing values, if any, may be replaced or merged. For example, for the server at address `10.1.2.3`, to replace existing client classes with `allow3` in the example above, write:

```
startAddr=10.1.2.3:allowClientClasses=allow3
```

To update only some of the values in a list use the notation += when specifying the attribute and value. In the example above, to allow a client class allow4 while keeping the `allow3`, write:

```
containerName=East: allowClientClasses+=allow4
```

Separate multiple attribute-value pairs with commas. For example, to change both the option set and the client classes for the server at `10.1.2.3`:

```
ipAddress=10.1.2.3:optionSet=OptionSet1,clientClasses=allow1|allow2|deny3
```

The configuration extension fields can directly contain text or can specify a file name. For example, to use the contents of the file `beginext.txt` as the extension at the beginning of the configuration file:

```
name=dhcp123.com.com:beginExtension=file:beginext.txt
```

V6 Addresses

To specify an IPV6 address, # must be used as the separator between the locator pairs and the modification pairs, since : is part of the address:

```
ipAddress=3FFE:0000:0000:0015::#clientClasses=allow1|allow2|deny3
```

ModifyDomainResourceRecord

Overview

The **ModifyDomainResourceRecord** CLI alters existing domain resource records in the system.

Usage

Direct

```
$INCHOME/jre/bin/java –cp $CLASSPATH
com.diamondip.ipcontrol.cli.ModifyDomainResourceRecord –u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ModifyDomainResourceRecord.sh –u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ModifyDomainResourceRecord.cmd –u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	Username
-p <pswd>	Yes	Password
-?	No	Print help
-f <update filename>	Yes	The name of the CSV file describing the modifications. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

```
$INCHOME/etc/cli/ModifyDomainResourceRecord.sh –u joe –p joepwd
-f updatedomainresourcerecs.txt -r updatedomainresourcerecs.reject
–e updatedomainresourcerecs.err
```

Output

If successful, the CLI updates the domain resource records per the input file and exits.

File Format

The **ModifyDomainResourceRecord** CLI uses attribute-value pairs to specify the records and fields to be changed. Each line in the input file must have a set of attribute-value pairs to locate the resource record to be changed and a second set specifies what fields to change and their new values.

The following table lists the available attributes for domain resource records and also indicates which can be used to locate a record.

Field	Attribute Name	Accepted Values	Locator Field/Required
Domain Name	domain	The name of the domain for this resource record.	Yes/Yes
Domain Type	domainType	The domain type of the domain. Defaults to "Default"	Yes/No
Owner	owner	The OWNER section of the resource record. Note that this section is specific to the type of resource record. Refer to the appropriate RFC for exact text that should be entered.	Yes/Yes
TTL	TTL	The Time to Live	No/No
Class	resourceRecClass	The value currently supported is IN. If not specified, defaults to IN.	Yes/No
Resource Record Type	resourceRecType	The type of resource record being updated.	Yes/Yes
Data	data	The text for the DATA area of the resource record. Note that this section is specific to the type of resource record. Refer to the appropriate RFC for exact text that should be entered.	Yes/No
Comment	comment	Text to be appended to the resource record.	No/No

Each input line specifies the locator attribute-value pairs, followed by a colon, followed by the modification attribute-value pairs. For example, to change the data and TTL for a record:

```
domain=40.10.in-addr.arpa.,domainType=Default,owner=10.10,
resourceRecClass=IN,resourceRecType=A:data="newDNS.ins.com", TTL=2400
```

This applies to the locator fields as well. For example, to change the owner for a record:

```
domain=40.10.in-
addr.arpa.,domainType=Default,owner=10.10,resourceRecClass=IN,resourceRecType=A:owner=30.10
```

To remove a value, specify the attribute but leave the value empty. For example, to remove the description from the above device:

```
domain=40.10.in-
addr.arpa.,domainType=Default,owner=10.10,resourceRecClass=IN,resourceRecType=A:comment=
```

Note on overlapping space:

If there are A records in overlapping space with identical owners, if the administrator's role does not indicate "Ignore" for "Allow Duplicate A Record (Owner) Checking", this CLI will fail with "Duplicate Resource Record".

V6 Addresses

To specify an IPV6 address, # must be used as the separator between the locator pairs and the modification pairs, since : is part of the address:

```
domain=example.com,domainType=Default,owner=router00015,data=3FFE:0000:0000:0015::,resource
RecClass=IN,resourceRecType=AAAA#TTL=2400
```

ModifyNetElementInterface

Overview

The **ModifyNetElementInterface** CLI alters existing Network Element Interfaces in the system.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.ModifyNetElementInterfaceCLI -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ModifyNetElementInterfaceCLI.sh -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ModifyNetElementInterfaceCLI.cmd -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	Username
-p <pswd>	Yes	Password
-?	No	Print help
-f <update filename>	Yes	The name of the CSV file describing the modifications. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

```
$INCHOME/etc/cli/ModifyNetElementInterfaceCLI.sh -u joe -p joepwd -f updatenei.txt
-r updatenei.reject -e updatenei.err
```

Output

If successful, the CLI updates Network Element Interfaces per the input file and exits.

File Format

The **ModifyNetElementInterface** CLI uses attribute-value pairs to specify the records and fields to be changed. Each line in the input file must have a set of attribute-value pairs to locate the Network Element Interface to be changed and a second set that specifies what attributes to change.

If an attribute is not included in the modifier set, then its value is not changed.

The following table lists the available attributes for Network Element Interfaces and also indicates which can be used to locate a record.

Field	Attribute Name	Accepted Values	Locator Field/ Required
Network Element Name	netElementName	The Name of the Network Element	Yes/Yes
Interface Name	interfaceName	The name of the interface	Yes/Yes
Status	status	The interface status. This can be one of "Disabled", "Enabled", or "Deployed".	No/No

Each input line specifies the locator attribute-value pairs, followed by a colon, followed by the modification attribute-value pairs. To leave an attribute unchanged, simply omit it from the modification attribute-value pairs. Separate multiple attribute-value pairs with commas. For example, to change a network element's interface name and status:

```
netElementName=RouterOne,interfaceName=Ethernet1:interfaceName=NewName,status=Enabled
```

ModifyPendingApproval

Overview

The **ModifyPendingApproval** CLI enables the approval or rejection of changes submitted to the administrator's pending approval queue.

Usage

Direct

```
$INCHOME/jre/bin/java –cp $CLASSPATH
com.diamondip.ipcontrol.cli.ModifyPendingApprovalCLI  –u <userId> -p <pswd>
-f < update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ModifyPendingApproval.sh –u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ModifyPendingApproval.cmd –u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	Username
-p <pswd>	Yes	Password
-?	No	Print help
-f <update filename>	Yes	The name of the CSV file containing modify instructions. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

```
$INCHOME/etc/cli/ModifyPendingApproval.sh –u joe –p joepwd -f pendingapprovals.csv
-r pendingapprovals.reject –e pendingapprovals.err
```

Output

If successful, the CLI approves and rejects changes per the input file and exits.

File Format

The **ModifyPendingApproval** CLI uses attribute-value pairs to specify the records and action to be taken. Each line in the input file must have an attribute-value pair to locate the pending approval record to be resolved followed by a set that specifies the action to be taken.

The following table lists the available attributes for this CLI.

Field	Attribute Name	Accepted Values	Locator Field
Workflow id	workflowId	The id of the pending approval request retrieved using an Export*Item*PendingApprovalCLI, for example, ExportResourceRecordPendingApproval.	Yes
Action	action	Specify "Approve" or "Reject". (required)	No
Reason	reason	Reason for rejection (optional)	No

Each input line specifies the locator attribute-value pair, followed by a colon, followed by the optional modification attribute-value pairs.

For example, to reject a pending approval:

```
workflowId=1018:action="Reject",reason="change not appropriate at this time"
```

ModifyPrefixPool

Overview

The **ModifyPrefixPool** CLI alters existing prefix pools in the system.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.ModifyPrefixPoolCLI -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ModifyPrefixPool.sh -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/ModifyPrefixPool.cmd -u <userId> -p <pswd>
-f <update filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	Username
-p <pswd>	Yes	Password
-?	No	Print help
-f <update filename>	Yes	The name of the CSV file containing modify instructions. See below for the required file format.
-r <rejects file>	No	The name of the file that rejected records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.

Usage Example

```
$INCHOME/etc/cli/ModifyPrefixPool.sh -u joe -p joepwd -f updateprefixpools.txt
-r updateprefixpools.reject -e updateprefixpool.err
```

Output

If successful, the CLI updates the prefix pools per the input file and exits.

File Format

The **ModifyPrefixPool** CLI uses attribute-value pairs to specify the records and fields to be changed. Each line in the input file must have a set of attribute-value pairs to locate the prefix pool to be changed and a second set specifies what fields to change and their new values.

The following table lists the available attributes for prefix pools and also indicates which can be used to locate a record.

ModifyPrefixPool

Field	Attribute Name	Accepted Values	Locator Field
Start Address	startAddr	The IP Address of the first address in the pool. This address must be in a block with an In-Use/Deployed status.	Required
Length	length	The length of the prefix pool	No
Address Pool Type	type	One of "Dynamic PD DHCPv6", "Automatic PD DHCPv6".	No
Name	name	Address Pool name. Defaults to "Start Address/Length"	No
Container	container	The name of the container that holds the block in which the pool is defined. This is required only if there is overlapping address space in use and the start address is in overlapping space. The container is then used to uniquely determine the block that will contain the address pool.	Yes, but not required unless Start address is not unique.
Delegated Prefix Length	delegatedPrefixLength	Length of the delegated prefixes.	No
Shortest Prefix Length	shortestPrefixLength	Shortest possible prefix to delegate. CNR only.	No
Longest Prefix Length	longestPrefixLength	Longest possible prefix to delegate. CNR only.	No
Primary Net Service	primaryNetService	The name of the DHCP server that will serve addresses from this pool	No
DHCP Option Set	dhcpOptionSet	The name of an Option Set used with this pool.	No
DHCP Policy Set	dhcpPolicySet	The name of a Policy Set used with this pool.	No
Allow DHCP Client Classes	allowClientClasses	A list of Client Classes that are allowed in this prefix pool. Separate the list entries with a vertical bar "|".	No
Deny DHCP Client Classes	denyClientClasses	A list of Client Classes that are NOT allowed in this prefix pool. Separate the list entries with a vertical bar "|".	No
Overlap Interface IP	overlapInterfaceIp	Flag to allow a DHCPv6 pool to overlap an interface address. This flag may be set only if pool is managed by a CNR DHCPv6 server. This is ignored for DHCPv4 pools.	No

Each input line specifies the locator attribute-value pairs, followed by a pound sign, followed by the modification attribute-value pairs. For example, to change the length for the pool starting at address `2001:db8:0:1::`

```
startAddr=2001:db8:0:1::#length=66
```

Separate multiple attribute-value pairs with commas. For example, to change both the length address and the DHCP Server for the address pool starting at `2001:db8:0:1::`

```
startAddr=2001:db8:0:1::#length=66,primaryNetService=newserver
```

This applies to the locator fields as well. For example, to change the length for the pool starting at `2001:db8:0:1::` in Container `Private`:

```
startAddr=2001:db8:0:1::,container=Private#length=66
```

Some values are lists. Separate the list elements with a vertical bar. For example, to allow two Client Classes for the pool starting at `2001:db8:0:1::`

```
startAddr=2001:db8:0:1::#allowClientClasses=allow1|allow2
```

For fields that are lists, existing values, if any, may be replaced or merged. For example, for the pool starting at `2001:db8:0:1::`, to replace `allow1` and `allow2` with `allow3` in the example above write:

```
startAddr=2001:db8:0:1::#allowClientClasses=allow3
```

To update only some of the values in a list use the notation += when specifying the attribute and value. In the example above to allow a client class `allow4` while keeping the `allow3`, write:

```
startAddr=2001:db8:0:1::#allowClientClasses+=allow4
```

To remove a value, specify the attribute but leave the value empty. For example, to remove the `allowClientClasses` from the above pool:

```
startAddr=2001:db8:0:1::#allowClientClasses=
```

SplitBlock

Overview

The **SplitBlock** CLI allows the user to split an existing block into smaller blocks. This CLI allows you to specify a single block on the command line. Note that splitting a block attached to multiple containers is not supported via the CLI.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.SplitBlockCLI -u <userId> -p <pswd>
-b <block Name> [-c <container name>] [-t <target start address>] [-s <target size>]
[-q <equal sizes>] [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/SplitBlockCLI.sh -u <userId> -p <pswd>
-b <block Name> [-c <container name>] [-t <target start address>] [-s <target size>]
[-q <equal sizes>] [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/SplitBlockCLI.cmd -u <userId> -p <pswd>
-b <block Name> [-c <container name>] [-t <target start address>] [-s <target size>]
[-q <equal sizes>] [-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	Username
-p <pswd>	Yes	Password
-?	No	Print help
-r <rejects file>	No	The name of the file that rejected (non-deleted) records will be placed in.
-e <error messages>	No	The name of the file that error messages will be reported in.
-b <block name>	Yes	The name of the block to be split. This is typically the CIDR address, e.g. 10.1.2.0/24. However, if the block name was set manually during allocation, that value must be used instead.
-c <container name>	Must be specified if the block name is not unique.	The name of the container holding the block. If this parameter is supplied, the container is searched for the block to join. Otherwise, the whole system is searched.
-t <target start address>	No	The start address of the target block. This is useful for creating a block using the specified start address and target block size. If no start address is specified, the start address of the block being split will be used.
-s <target size>	Yes	The desired CIDR block size after the split. This parameter works in conjunction with the “equalSizes” parameter.
-q <equal sizes>	No	Specify true or false. If true, the block is split such that all resulting blocks have the “target size” CIDR size. If false, the block is split such that the fewest number of new blocks is created, along with two blocks of “targetSize”. The default is false.

Usage Example

This example splits the block `10.1.2.0/24` into 8 `/27` blocks. If `-e` were false, the result would be one `/25`, one `/26` and two `/27` blocks.

```
$INCHOME/etc/cli/SplitBlock.sh -u joe -p joepwd -b 10.1.2.0/24 -s 27 -q true
```

UseNextReservedIPAddress

Overview

The **UseNextReservedIPAddress** CLI is used to mark the next reserved IP Address in the specified block, for the specified device type, to in-use. The block must have a status of "In Use/Deployed". Within this block, there should be a range of addresses with a type of "Reserved" and a status of "reserved" for the given device type. This CLI should not be used with address pools with a status of "reserved". The next lowest or highest IP address within the range will be assigned a type of "Static" and a status of "in-use". If a hostname is specified, it will be applied to the device associated with the IP Address. In addition, there is an option to add resource records for the device.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.UseNextReservedIPAddress -u <userId> -p <pswd>
-b <blockaddress> -d <devicetype> [-h <hostname>] [-r <rr flag>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/UseNextReservedIPAddress.sh -u <userId> -p <pswd>
-b <blockaddress> -d <devicetype> [-h <hostname>] [-r <rr flag>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/UseNextReservedIPAddress.cmd -u <userId> -p <pswd>
-b <blockaddress> -d <devicetype> [-h <hostname>] [-r <rr flag>] [-?]
```

Parameters

Parameter	Required	Description
-u <Username>	Yes	Username
-p <pswd>	Yes	Password
-?	No	Print help
-b <block address>	Yes	The address of an "In Use/Deployed" block containing "reserved" type addresses, not in address pools, of the device type specified in the –d parameter.
-d <device type>	Yes	Device type of address to mark in-use.
-h <host name>	No	If specified, will be applied to the device associated with the IP Address.
-r <resource record flag>	No. Defaults to "false".	Specify "true" or "false". When "true", resource records will be added for the device.

Utilities

DhcpRelease

Overview

Please note that the DhcpRelease CLI is applicable only for DHCPv4 environments.

The **`DhcpRelease`** CLI is used to force the release of a DHCP lease. A DHCP lease is a contract for the use of an IP address between the DHCP server and client for a specific amount of time. This CLI can be used in lieu of performing a release of the lease from the client itself. Instead, this CLI will create a DHCP Release packet identifying the client's hardware address and IP address, and send the request to the DHCP server. The DHCP server will then free the lease as if the release was sent from the actual client. This then makes the freed IP address available for lease assignment to another client on the subnet.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH
com.diamondip.ipcontrol.cli.DhcpReleaseCLI [-s <server>] -m <macaddr> -i <ipaddr>
[-f <filename>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DhcpRelease.sh [-s <server>] -m <macaddr> -i <ipaddr>
[-f <filename>] [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/cli/DhcpRelease.cmd [-s <server>] -m <macaddr> -i <ipaddr>
[-f <filename>] [-?]
```

Parameters

Parameter	Required	Description
-s <server>	No	DHCP Server IP Address (default = 127.0.0.1)
-m <macaddr>	Yes	Client MAC address formatted as hexadecimal characters separated by colons – e.g. a1:b2:c3:d4:e5:f6
-i <ipaddr>	Yes	The IP address to be released by the DHCP server. This IP address should be associated with a lease for the client identified by the 'macaddr' parameter.
-?	No	Print help
-f <filename>	No	The name of the CSV file which defines leases to be released. See below for the required file format.

Usage Examples

This example releases the lease for IP address 10.10.10.101 for the client with MAC address `de:ad:be:ef:ca:fe` on the DHCP server at IP address `10.10.10.1`.

```
$INCHOME/etc/cli/DhcpRelease.sh -s 10.10.10.1 -m de:ad:be:ef:ca:fe -i 10.10.10.101
```

This example issues a release for each lease identified in the *dhcprelease.csv* file.

```
$INCHOME/etc/cli/DhcpRelease.sh -f dhcprelease.csv
```

Output

The output from the CLI indicates the specific parameters used to issue the DHCP Release. These log entries can be found in the CLI logfile (*$INCHOME/etc/cli/log/ns_webservice.log*) when the appropriate logging level is configured:

```
16:10:02,506 ()[main] INFO  DHCPRelease - Building DHCP Release packet
16:10:02,606 ()[main] INFO  DHCPRelease - Sending DHCP Release to server=10.10.10.1 for
MAC=de:ad:be:ef:ca:fe and IP=10.10.10.101
```

If successful, a corresponding entry for the DHCP release appears in the appropriate syslog output file on the DHCP server.

File Format

Col	Field	Accepted Values	Required
A	Server IP Address	The IP Address of the DHCP Server	Yes
B	Client MAC Address	The MAC Address of the client Format: a1:b2:c3:d4:e5:f6	Yes
C	Client IP Address	The IP Address to be released	Yes

Notes

This CLI should be used with *extreme caution*. Only authorized network administrators should attempt to run this CLI. A lease is a contract between a DHCP server and a DHCP client. Using this CLI to simulate client behavior is potentially dangerous. For example, releasing an active lease for a client that is still on the network could lead to duplicate IP address situations. ***Therefore, this CLI should only be used when it can be guaranteed by the network administrator that the lease(s) are no longer in use by the client***. Due to the different DHCP client implementations, the results of releasing an active lease are unpredictable. If the client is actively using the IP address when the lease is released, the client may immediately loose network connectivity. If the client maintains the lease offline when the lease is released, then the client may not be able to obtain network connectivity on the subnet for which the lease was released.

Caveats

Reliability

It is important to note that there is no reply message from the DHCP server to the client in response to the DHCP Release. Therefore, the CLI cannot verify that the release was successful. In addition, the packet is sent via UDP, making it impossible to detect if the server even received the release message.

Network Connectivity

This CLI must be run from a network host which allows DHCP traffic to the DHCP server. Specifically, the CLI will open a high numbered (1025+), ephemeral, port on the local host and direct UDP packets to port 67 on the specified DHCP server. Router and firewall rules must allow such traffic.

Localhost Usage

In many cases, to avoid network connectivity issues noted above, it is convenient to run the CLI from the same host as the DHCP server and specify a server IP address of `127.0.0.1`. Each IPAM platform has different behavior with respect to running the CLI locally.

- **Windows** – no known issues.
- **Linux** – supported only with ISC DHCP v3.x. ISC DHCP v4.x does not support listening on the loopback address.

Linux kernel versions 2.6.18-2.6.26, inclusive, contain a bug that creates a bad UDP checksum for packets sent on the localhost. If you are running a Linux distribution with a kernel version in this range, the **`DhcpRelease`** CLI will not function from the localhost. You can verify your kernel version by running `uname -a` in a console window.

Additionally, the server must be configured with the loopback subnet in the *dhcpd.conf* file. By default, the appropriate statement is automatically added to the dhcpd.conf file when a DHCP Configuration task is performed.

```
subnet 127.0.0.1 netmask 255.255.255.255 {
}
```

If the server's configuration does not include this statement, modify the DHCP server's configuration via Topology → Network Services → Edit DHCP Server. Select the Extensions tab and insert the above loopback subnet declaration to be appended to the end of the configuration file. After making these changes, push the new configuration to the DHCP server via Management → Configuration/Deployment → DHCP Configuration – All Files.

Once the server has been restarted using the modified `dhcpd_start` script and is configured with the loopback subnet, then the **`DhcpRelease`** CLI can be run from the localhost of the DHCP server itself.

Purge

Overview

The Purge utility will delete records from the set of IPAM tables listed below that are older than the specified date constraint (-c or -d) in batches of 1000 records.

To configure the date constraint from which all older records will be deleted, specify a date (-c) or number of days from today (-d). NOTE: a date constraint is required to operate the utility.

To limit the tables to purge, use the -t parameter which is constrained to the following tables:

- ADDRPOOLHISTORY
- ALERTLOG
- AUDITLOG
- BLOCKHISTORY
- CONTAINERHISTORY
- EVENTLOG

If the performance of the individual delete statements becomes an issue, the batch size can be changed using the -b parameter.

Database connection information is automatically derived from the IPAM jdbc.properties found in the product CLASSPATH.

Usage

Direct

```
$INCHOME/jre/bin/java -cp $CLASSPATH com.diamondip.ipcontrol.cli.purge
[-p] [-v] [-i] [-q] [-b size] [-t table,...] -c YYYY-MM-DD | -d days [-?]
```

Via command script (Unix)

```
$INCHOME/etc/support/purge.sh
[-p] [-v] [-i] [-q] [-b size] [-t table,...] -c YYYY-MM-DD | -d days [-?]
```

Via command script (Windows)

```
%INCHOME%/etc/support/purge.cmd
[-p] [-v] [-i] [-q] [-b size] [-t table,...] -c YYYY-MM-DD | -d days [-?]
```

Parameters

Parameter	Required	Description
-p	No	Preview mode, writes dry run report to stdout.
-v	No	Verbose mode, writes verbose output to stdout.
-i	No	Interactive mode, prompts user for SQL execution permission.

-q	No	Quiet mode, suspends user prompting and progress reporting.
-b <size>	No	Overwrites default batch size (1000).
-c <YYYY-MM-DD>	Yes, if –d not specified.	Purge date. All records older than this date will be deleted.
-d <days>	Yes, if –c not specified.	Number of days from the current date to keep. All older records will be deleted.
-t <table,...>	No	Comma separated list of tables to purge.
-?	No	Print help

Usage Example

This example will delete all records older then 2013-05-01 in a batch size of 5 for the addrpoolhistory table.

```
$INCHOME/etc/support/purge.sh -b 5 -c 2013-05-01 -t addrpoolhistory
```

```
-----------------------------------------------
IPControl DB Purge CLI - Copyright BT Diamond IP
Version: v0.2 (2013)
Runtime: Wed May 15 15:54:55 EDT 2013

[ Utility Property File Summary ]
+ Logging properties file = C:/bin/Cisco Prime Network Registrar IPAM
/etc/support/purge_log4j.properties
+ JDBC properties file    = C:/bin/Cisco Prime Network Registrar IPAM
/jdbc.properties

[ JDBC Properties ]
+ jdbc.driverClassName    = oracle.jdbc.driver.OracleDriver
+ jdbc.url                = jdbc:oracle:thin:@localhost:1521:ipam
+ jdbc.username           = incadmin4
+ jdbc.password           = ********

[ Job Properties ]
+ Preview                 = false
+ Verbose                 = false
+ Interactive             = false
+ Quiet Mode              = false
-----------------------------------------------
2013-05-15 15:54:56 -
2013-05-15 15:54:56 - #JOB# - Execute purge ADDRPOOLHISTORY by LOGDATE
2013-05-15 15:54:56 - + Purge recs older then: 2013-05-01
2013-05-15 15:54:56 - + Batch size: 5
2013-05-15 15:54:56 - + Recs found: 33
2013-05-15 15:54:56 - 33 ADDRPOOLHISTORY records will be deleted in 7 batches.
2013-05-15 15:54:56 - + DELETESQL = DELETE FROM ADDRPOOLHISTORY WHERE LOGDATE <
TO_DATE('2013-05-01', 'YYYY-MM-DD')  AND ROWNUM <= 5
2013-05-15 15:54:56 - Batch 1 deleted 5 recs
2013-05-15 15:54:56 - Batch 2 deleted 5 recs
2013-05-15 15:54:56 - Batch 3 deleted 5 recs
2013-05-15 15:54:56 - Batch 4 deleted 5 recs
2013-05-15 15:54:56 - Batch 5 deleted 5 recs
2013-05-15 15:54:56 - Batch 6 deleted 5 recs
2013-05-15 15:54:56 - Batch 7 deleted 3 recs
2013-05-15 15:54:56 - #JOB# - Purge complete, deleted 33 recs, 16ms
2013-05-15 15:54:56 -
2013-05-15 15:54:56 - #DONE# - Processing completed, 17ms
```

Output

Unless quiet mode (-q) in invoked the utility will write out its operating environment parameters and a running summary of its execution results to stdout. A full execution report will also be logged to the location specified in $INCHOME/etc/support/purge_log4j.properties for every run. By default the log will be found under ${INCX_HOME}/ etc/support /log/purge.log.

Notes

This utility should be used with caution as deleted records can not be restored without a database backup. For this reason it's recommended to first run the purge utility in preview mode (e.g. $INCHOME/etc/support/purge.sh **-p**) to get an idea of the scope of the work that will be done, followed by interactive mode (e.g. $INCHOME/etc/support/purge.sh **-i**) to force a user to confirm each table aggregate delete operation.

Application Program Interfaces (API)

Using the API

IPAM provides its API as a set of Web Services. Invoke the API by implementing web service clients, using the client technology of your choice.

The interfaces are grouped into Imports, Gets, Tasks, Exports, Updates and Deletes. Each service is explained in detail, including the WSDL applicable to each interface, in the sections that follow.

To view the complete WSDL for each of the services, see:

Imports:
http://localhost:8080/inc-ws/services/Imports?wsdl

Gets:
http://localhost:8080/inc-ws/services/Gets?wsdl

Tasks:
http://localhost:8080/inc-ws/services/TaskInvocation?wsdl

Exports:
http://localhost:8080/inc-ws/services/Exports?wsdl

Updates:
http://localhost:8080/inc-ws/services/IncUseNextReservedIPAddress?wsdl

Deletes:
http://localhost:8080/inc-ws/services/Deletes?wsdl

Invoking the web service and authentication

IPAM uses HTTP Basic Authentication (BASIC_AUTH), using authentication handlers that are invoked before the web service request is called. In order to use web services, the client must pass the IPAM login name and password. IPAM will then validate that this is a valid combination, and that the administrator is authorized to use web services (via the Allow Command Line Interface Access checkbox on the Administrator Policies screen).

Below is a sample code fragment of a client using Java and Axis to invoke the **`importDevice`** operation of the Imports web service. Note that the stubs used in the example are generated by Axis' wsdl2Java tool. For more information on Axis and wsdl stubs, see
http://ws.apache.org/axis/java/user-guide.html

```
// Axis-generated class
```

```
ImportsServiceLocator locator = new ImportsServiceLocator();

// Setup for authorization handlers
ImportsSoapBindingStub stub = (ImportsSoapBindingStub)locator.getImports(url);
stub.setUsername("incadmin");
stub.setPassword("incadmin");

// Set up input parameter structure
WSDevice wsdevice = new WSDevice();
wsdevice.setIpAddress(blockName);
wsdevice.setDeviceType(deviceType);
wsdevice.setHostname(hostName);
wsdevice.setResourceRecordFlag(resourceRecordFlag);

// Use Axis-generated class to call the web service
String address = stub.importDevice(wsdevice);

// Error handling shown in next section
return address;
```

Below is an example of a client using .NET to invoke the findNetService operation of the Exports web service:

```
Public Shared Sub findNetService()
   Dim myCred As New NetworkCredential("incadmin", "incadmin")
   Dim myWS As New localhost.ExportsService()
   Dim myNS As localhost.WSNetService()
   Dim i As Integer

   myWS.Credentials = myCred
   myWS.Url = "http://localhost:8081/inc-ws/services/Exports"

   Try
     myNS = myWS.findNetService("", "", "", "", "", "", "")
       For i = 0 To myNS.Length - 1
         Console.WriteLine("Name={0} IP={1} Type={2}",
            myNS(i).name, myNS(i).ipAddress, myNS(i).type)
     Next
   Catch myErr As SoapException
            dumpError(myErr)
   Catch otherErr As Exception
            Console.WriteLine(otherErr.ToString)
   End Try
End Sub
```

Error Processing

When the web service finds an error during processing, it uses the fault codes defined in SOAP 1.1, along with additional information in the detail element, to convey the nature of the error. Below is a sample of the XML that will be sent to the client.

```
<soapenv:Envelope
     xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
     xmlns:xsd=http://www.w3.org/2001/XMLSchema
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
  <soapenv:Fault>
    <faultcode>soapenv:Server</faultcode>
    <faultstring>Unknown root block type: 0</faultstring>
    <faultactor>http://localhost:8080/nc/services/RootBlockImport</faultactor>
    <faultDetail>
      <returnCode>-14</returnCode>
    </faultDetail>
  </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

Java clients can catch the error as an Exception and access the message. Below is a sample code fragment of a client using Java and Axis to retrieve the information in the detail element of the XML:

```
try {
    Invoke web service
}
catch (Exception ex) {
    String errMsg = "Line " + inputLine + ": " +ex.getMessage();
    System.err.println("Exception - " + errMsg);
    if (ex instanceof AxisFault) {
        // Retrieve Axis Fault detail
        AxisFault fault = (AxisFault) ex;
        Element[] elements =fault.getFaultDetails();
        System.err.println("AxisFault returned:");
        System.err.println("  faultcode: " + fault.getFaultCode());
        System.err.println("  faultstring: "+fault.getFaultString());
        System.err.println("  faultdetail tag: " + elements[0].getTagName());
        System.err.println("  faultdetail node value: " +
            elements[0].getFirstChild().getNodeValue());
    } else {
        ex.printStackTrace();
    }
}
```

Other SOAP toolkits (e.g., .NET and Perl) can parse the XML for the details, or ignore those tags if that level of detail about the exception is not required for the application.

Available Application Program Interface Matrix

Object	Import	Modify	Delete	Export
Address Pool	X	X	X	
Administrator	X	*(see import)*	X	X
Administrator Role	X	*(see import)*	X	X

Object	Import	Modify	Delete	Export
Aggregate Block	X		X	
Block	X	X	X	X
Container	X	X	X	X
Device	X	X	X	X
Device Interface			X	
Device RR	X	X	X	X
DHCP Server	X	X		
Domain	X		X	
Domain RR	X	X	X	X
Galaxy Domain	X			
Network Element	X	*(see Import)*	X	X
NetElementInterface	X	X	X	
Netservice	*(see ImportDHCP Server)*		X	X
Network Link	X	*(see Import)*	X	X
PrefixPool	X	X	X	X
RootBlock	X		X	
Zone	X	*(see ImportZone)*	X	
Zone RR	X		X	
Next Available IP	X			
JoinBlock		X		
Site (multiple block allocation)	X	N/A	N/A	N/A
Split Block		X		
Detach Block		X		

Task	Import	Modify	Delete	Export
GlobalNetElementSync	X		X	X
GlobalNetServiceSync	X		X	X
ImportElementSnapshot				
ImportServiceSnapshot				

GlobalRollup	X		X	
DiscoverNetElement	X		X	
DhcpUtilization	X		X	
GetTask	X			
GetTask Status	X			

Imports

Overview

This section explains the web services available for importing information to IPAM. Each of these services is available as an operation in the Imports web service. You can see the complete WSDL at: http://localhost:8080/inc-ws/services/Imports?wsdl

AddSite

Overview

The **AddSite** API enables the web service client to add a site to a container using an existing Site Template.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **addSite** request and response messages.

```
< wsdl:message name="addSiteResponse">
    <wsdl:part name="addSiteReturn" type="soapenc:string"/>
</wsdl:message>
< wsdl:message name="addSiteRequest">
    <wsdl:part name="site" type="tns2:WSSite"/>
</wsdl:message>
```

Response

The string returned in the response contains a comma-separated list of block names added, for example, "`10.0.0.0/24, 10.0.1.0/25`".

Request

The complex type **WSSite**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSSite

Below is the portion of Imports.wsdl that describes **WSSite**, the parameter structure passed to **addSite**. The elements are described in the table that follows.

```
<complexType name="WSSite">
  <sequence>
    <element name="container" nillable="true" type="soapenc:string"/>
    <element maxOccurs="unbounded" name="siteBlockDetails" nillable="true"
      type="tns2:WSSiteBlockDetails"/>
    <element name="siteTemplateName" nillable="true" type="soapenc:string"/>
  </sequence>
 </complexType>
```

Element	Description and accepted values	Required	Return Code	Faultstring
container	The name of the container in which to create the site. Names can be in either short or long format. Short format example: Dallas Long format example: InControl/Texas/Dallas Long format eliminates ambiguity in cases where there are duplicate container names.	yes	-42 -5	Container required Container *name* not found
siteBlockDetails	A repeating structure of parameters used in creating the blocks from the template. See below for details.	no	-214	Block details do not match site template details
siteTemplateName	The name of the site template to be used in creating the site.	yes	-211 -212 -213 -215	Site template name required Site template name not found Site template type does not match container type (logical/device) *type* exception applying site template: *message*

WSSiteBlockDetails

Below is the portion of Imports.wsdl that describes **`WSSiteBlockDetails`**, included in **`WSSite described above`**. The site block details must be specified in the order matching the sequence of the Site Template Detail records in the IPAM GUI. The elements are described in the table that follows.

```
<complexType name="WSSiteBlockDetails">
  <sequence>
    <element maxOccurs="unbounded" name="addrDetails" nillable="true"
      type="tns2:WSAllocationTemplateDetails"/>
    <element name="allocationReason" nillable="true" type="soapenc:string"/>
    <element name="allocationReasonDescription" nillable="true" type="soapenc:string"/>
    <element name="interfaceName" nillable="true" type="soapenc:string"/>
    <element name="swipName" nillable="true" type="soapenc:string"/>
    <element name="userDefinedFields" nillable="true"
      type="impl:ArrayOf_soapenc_string"/>
  </sequence>
</complexType>
```

Element	Description and accepted values	Required	Return Code	Faultstring
addrDetails	Define attributes of any address allocations of the allocation template specified by the site allocation template for this block. See below for syntax.	no		
allocationReason	The name of a pre-existing Allocation Reason. If Allocation Reason is not currently in IPAM, this field is skipped.	no	-22	Invalid allocation reason *allocReason*
allocationReasonDescription	A description of the reason for the allocation. Wrap the statement in "quotes" if it contains any commas.	no		
interfaceName	The target interface name. This is a locator field.	yes, for a device container	-19 -20	No interface found Interface name is required for device containers
swipName	SWIP name	yes, if required for this container/blocktype	-66 -70	SWIPname is required for this container/blocktype SWIPname is not allowed for this container/blocktype
userDefinedFields	Array of user defined fields. Each element in the array has the format "name=value" where "name is the UDF tag name.	yes, if required by template	-63 -61	Invalid UDF: *udf* Missing required UDF: *udf*

WSAllocationTemplateDetails

Below is the portion of Imports.wsdl that describes **`WSAllocationTemplateDetails`**, included in **`WSSiteBlockDetails described above`**. The elements are described in the table that follows.

```
<complexType name="WSAllocationTemplateDetails">
  <sequence>
    <element name="netserviceName" nillable="true" type="soapenc:string"/>
    <element name="offsetFromBeginningOfSubnet" type="xsd:boolean"/>
    <element name="~~sharename~~" nillable="true" type="soapenc:string"/>
    <element name="startingOffset" type="xsd:long"/>
  </sequence >
</complexType>
```

Element	Description and accepted values	Required	Return Code	Faultstring
netserviceName	The name of the network service for this address allocation.	no	-185	Invalid network service name: *name*
offsetFromBeginningOfSubnet	Specify **true** or **false**. This must match the specification in the Allocation Template.	yes		
~~sharename~~ *Deprecated*	~~The name used to link address pools together.~~	no		
startingOffset	Identify the address allocation within the template. This must match the specification in the Allocation Template.	yes	-173 -174	Block details do not match site template details Invalid starting offset: *offset* for allocation template: *template name*

DetachBlock

Overview

The **DetachBlock** API enables the web service client detach blocks from device containers in IPAM. If an Interface Address is specified and the block has multiple interface addresses on the specified container, then only the specified Interface Address will be detached.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **detachBlock** request and response messages.

```
<wsdl:message name="detachBlockResponse ">
  <wsdl:part name="detachBlockReturn" type="soapenc:string" />
</wsdl:message>
<wsdl:message name="detachBlockRequest">
  <wsdl:part name=" name="childBlock" type="tns1:WSChildBlock" />
</wsdl:message>
```

Response

The string returned in the response contains the name of the block detached, for example, `10.0.0.128/28`.

Request

The complex type **WSChildBlock**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSChildBlock

Below is the portion of Imports.wsdl that describes **WSChildBlock**, the parameter structure passed to **detachBlock**. The elements are described in the table that follows.

```
<complexType name="WSChildBlock">
  <sequence>
    <element name="SWIPname" nillable="true" type="xsd:string" />
    <element name="allocationReason" nillable="true" type="xsd:string" />
    <element name="allocationReasonDescription" nillable="true" type="xsd:string" />
    <element name="allocationTemplate" nillable="true" type="xsd:string" />
    <element name="blockAddr" nillable="true" type="xsd:string" />
    <element name="blockName" nillable="true" type="xsd:string" />
    <element name="blockSize" nillable="true" type="xsd:string" />
    <element name="blockStatus" nillable="true" type="xsd:string" />
    <element name="blockType" nillable="true" type="xsd:string" />
    <element name="container" nillable="true" type="xsd:string" />
    <element name="createReverseDomains" nillable="true" type="xsd:string" />
    <element name="description" nillable="true" type="xsd:string" />
    <element name="domainType" nillable="true" type="xsd:string" />
    <element name="interfaceAddress" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="interfaceName" nillable="true" type="xsd:string" />
    <element name="ipv6" type="xsd:boolean"/>
    <element name="primarySubnet" type="xsd:boolean"/>
    <element name="nonBroadcast" type="xsd:boolean"/>
    <element name="userDefinedFields" nillable="true" type="impl:ArrayOf_xsd_string" />
    <element name="excludeFromDiscovery" nillable="true" type="xsd:string" />
  </sequence>
```

```
</complexType>
```

Element	Description and accepted values	Required	Return Code	Faultstring
SWIPname	Ignored	no		
allocationReason	Ignored	no		
allocationReason Description	Ignored	no		
allocationTemplate	Ignored	no		
blockAddr	The address of the block to detach	**yes**, if blockName is not specified	-183 -26 -36	Blockname or address space/block size required Invalid IpAddress: *blockAddr* Block *blockAddr* not found
blockName	The name of the block to detach	**yes**, if blockAddr is not specified	-36	Block *blockName* not found
blockSize	The size of the block in short-notation (e.g., 24 for a 255.255.255.0 network).	**yes**, if blockAddr is used	-15	Invalid block size: *blockSize*
blockStatus	Ignored	no		
blockType	Ignored	no		
container	The name of the container from which to detach block. Names can be in either short or long format. Short format example: **Dallas**. Long format example: **InControl/Texas/Dallas**. Long format eliminates ambiguity in cases where there are duplicate container names.	yes	-42 -5 -184	Container required Container *container* not found Block is only attached to one container. Use delete.
createReverseDomains	Ignored	no		
description	Ignored	no		
domainType	Ignored	no		
interfaceAddress	The specific address(es) for the interface IP address(es) to be detached.	no		
interfaceName	Ignored	no		
ipv6	Ignored	no		
primarySubnet	Ignored	no		
userDefinedFields	Ignored	no		
excludeFromDiscovery	Ignored	no		

ImportAddressPool

Overview

The **ImportAddressPool** API enables the web service client to import or modify address pools in IPAM.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **importAddressPool** request and response messages.

```
<wsdl:message name="importAddressPoolRequest">
  <wsdl:part name="addrpool" type="tns2:WSAddrpool"/>
</wsdl:message>
<wsdl:message name="importAddressPoolResponse"/>
```

Response

There is no data in the response.

Request

The complex type **WSAddrpool**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSAddrpool

Below is the portion of *Imports.wsdl* that describes **WSAddrpool**, the parameter structure passed to **importAddressPool**. The elements are described in the table that follows.

```
<complexType name="WSAddrpool">
  <sequence>
    <element name="allowClientClasses" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="container" nillable="true" type="soapenc:string"/>
    <element name="denyClientClasses" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="dhcpOptionSet" nillable="true" type="soapenc:string"/>
    <element name="dhcpPolicySet" nillable="true" type="soapenc:string"/>
    <element name="endAddr" nillable="true" type="soapenc:string"/>
    <element name="failoverNetService" nillable="true" type="soapenc:string"/>
    <element name="id" nillable="true" type="soapenc:string"/>
    <element name="name" nillable="true" type="soapenc:string"/>
    <element name="overlapInterfaceIp" type="xsd:boolean"/>
    <element name="prefixLength" nillable="true" type="soapenc:string"/>
    <element name="primaryNetService" nillable="true" type="soapenc:string"/>
    <element name="sharename" nillable="true" type="soapenc:string"/>
    <element name="startAddr" nillable="true" type="soapenc:string"/>
    <element name="type" nillable="true" type="soapenc:string"/>
  </sequence>
</complexType>
```

Element	Description and accepted values	Required	Return Code	Faultstring
id	The internal identifier for this address pool object. If this is not set, a new address pool is created. If this is set, the address pool with the matching identifier is updated.	No for creates, Yes for updates.	-117	Addrpool ID=<id> not found Invalid ID <id> on addrpool object
startAddr	The IP Address of the first address in the pool. This address must be in a block with an In-Use/Deployed status.	Yes	-115 -36 -97 -115	Missing Start or End IP Address Block Not Found Block Not Unique Invalid Start Address
endAddr	The IP Address of the last address in the pool. This address must be in the same block as the Start Address. In addition, the Start and End addresses must not overlap any other pools.	Yes for IPv4 pools.	-115 -26 -26 -115	Missing Start or End IP Address End Address outside of block End address must be after Start Address Invalid End Address
type	One of "Dynamic DHCP", "Automatic DHCP", "Manual DHCP", "Static", "Reserved". "Dynamic NA DHCPv6", "Automatic NA DHCPv6", "Dynamic TA DHCPv6", "Automatic TA DHCPv6"	Yes	-73	Invalid Address Type
name	Address Pool name. Defaults to "Start Address-End Address"	No		
~~sharename~~ *Deprecated*	~~The name used to link address pools together.~~	No		
container	The name of the container that holds the block in which the pool is defined. This is required only if there is overlapping address space in use and the start address is in overlapping space. The container is then used to uniquely determine the block that will contain the address pool.	No, unless startAddr is not unique.	-36	Block not found in container

Element	Description and accepted values	Required	Return Code	Faultstring
primaryNet Service	The name of the DHCP server that will serve addresses from this pool. Note: This is required when a DHCP server is not defined for the subnet, and this address pool is one of the DHCP address types: "Dynamic DHCP", "Automatic DHCP", "Dynamic NA DHCPv6", "Automatic NA DHCPv6", "Dynamic TA DHCPv6", "Automatic TA DHCPv6"	See note	-94	DHCP Server not found
failoverNet Service	The name of the failover DHCP server that will serve addresses from this pool. To use this field, primaryNetService must also be specified.	No	-94 -96	DHCP Server not found Failover not valid without a primary
dhcpOptionSet	The name of an Option Set used with this pool. For IPV4 address pools, the DHCP option set applies only to non-CNR DHCP servers. For IPV6 address pools, the DHCP option set applies only to CNR DHCP servers.	No	-67	DHCP Option Set not found
dhcpPolicySet	The name of a Policy Set used with this pool. For IPV4 address pools, the DHCP policy set applies only to non-CNR DHCP servers. For IPV6 address pools, the DHCP policy set applies only to CNR DHCP servers.	No	-68	DHCP Policy Set not found
allowClient Classes	For "Dynamic DHCP" and "Automatic DHCP" type pools: An array of Client Classes that are allowed in this address pools. Each element of the array names a different Client Class.	No	-116	DHCP Client Class not found
denyClientClas ses	For "Dynamic DHCP" and "Automatic DHCP" type pools: An array of Client Classes that are NOT allowed in this address pools. Each array element names a different Client Class.	No	-116	DHCP Client Class not found
prefixLength	CIDR size of an IPv6 pool. This element is ignored for an IPv4 address pool.	Yes for IPv6 pools.	-258	Missing Prefix length.

Element	Description and accepted values	Required	Return Code	Faultstring
overlapInterfaceIp	Flag to allow a DHCPv6 pool to overlap an interface address. This flag may be set only if pool is managed by a CNR DHCPv6 server. This is ignored for DHCPv4 pools.	No	-344	Invalid DHCP server.

ImportAdmin

Overview

The **ImportAdmin** API enables the web service client to import administrators to IPAM.

Note that while adminstrators of administrator type "NORMAL" can import and modify administrators and their roles, only MASTER administrators can import or update administrator policies and assignable roles. NORMAL administrators will receive an error message if policies or assignable roles are specified on import. On an import request to modify an administrator, if a NORMAL administrator specifies policies or assignable roles, that information is ignored.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **importAdmin** request and response messages.

```
<wsdl:message name="importAdminRequest">
  <wsdl:part name="wsAdmin" type="tns2:WSAdmin"/>
</wsdl:message>
<wsdl:message name="importAdminResponse">
  <wsdl:part name="importAdminReturn" type="soapenc:string"/>
</wsdl:message>
```

Response

The string returned in the response contains the login id of the administrator allocated.

Request

The complex type **WSAdmin**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSAdmin

Below is the portion of *Imports.wsdl* that describes **WSAdmin**, the parameter structure passed to importAdminRole.

```
<complexType name="WSAdmin">
    <sequence>
     <element name="address1" nillable="true" type="soapenc:string"/>
     <element name="address2" nillable="true" type="soapenc:string"/>
     <element name="address3" nillable="true" type="soapenc:string"/>
     <element name="adminType" nillable="true" type="soapenc:string"/>
     <element name="assignableRoles" nillable="true"
type="impl:ArrayOf_soapenc_string"/>
     <element name="authorizeExternally" type="xsd:boolean"/>
     <element name="email" nillable="true" type="soapenc:string"/>
     <element name="enabled" type="xsd:boolean"/>
     <element name="fax" nillable="true" type="soapenc:string"/>
     <element name="firstName" nillable="true" type="soapenc:string"/>
     <element name="id" nillable="true" type="soapenc:int"/>
     <element name="lastName" nillable="true" type="soapenc:string"/>
     <element name="loginId" nillable="true" type="soapenc:string"/>
```

```
     <element name="pager" nillable="true" type="soapenc:string"/>
     <element name="password" nillable="true" type="soapenc:string"/>
     <element name="phone" nillable="true" type="soapenc:string"/>
     <element name="policies" nillable="true" type="tns2:WSAdminPolicies"/>
     <element name="roles" nillable="true"
type="impl:ArrayOf_soapenc_string"/>
    </sequence>
</complexType>
```

Element	Description and accepted values	Required	Return Code	Faultstring
address1	Mailing address	no		
address2	Mailing address	no		
address3	Mailing address	no		
adminType	Specify one of **MASTER**, **NORMAL**, or **READONLY**. **MASTER** users have full control over the entire IPAM system. **NORMAL** users have ordinary read-write permissions, and may be restricted to working with only certain portions of the IPAM system. **READONLY** users have read-only access to IPAM, and may be restricted to seeing only certain portions of the IPAM system. Defaults to **READONLY**.	no	-297	Administrator type must be MASTER, NORMAL or READONLY
assignableRoles	Assignable Administrator Roles for this administrator. Assignable Roles are roles that the Administrator can assign to another Administrator.	no	-303	Role not found: *role*
authorizeExternally	Determines whether this user's username and password will be passed to an external authentication system. This is setup by the IPAM administrator in the System Policies screen. Accepted values are **true** or **false**. Defaults to **false.**	no		
email	Email address	no		
enabled	Determines whether this user can access the IPAM system. Accepted values are **true** or **false**.	no		
fax	Fax number	no		
firstName	First (given) name	no		
id	The internal ID of this administrator, as provided by a GetAdmin call. If this is set, the administrator is updated instead of added.	yes, for modify	-295	Admin ID= *id* not found

Element	Description and accepted values	Required	Return Code	Faultstring
lastName	Last (family) name	no		
loginId	Administrator login ID (for example, jsmith10)	yes	-293 -296	Admininistrator login id required Duplicate adminstrator login id:
pager	Pager number	no		
password	Administrator's password. (Import only; cannot be overwritten)	yes	-294	Administrator password required
phone	Phone number	no		
policies	A WSAdminPolicies structure, described in the ImportAdminRole section of this API chapter.	no		
roles	Administrator Roles for this administrator.	no	-303	Role not found: *role*

`WSAdmin` contains the parameter structure **`WSAdminPolicies`**. The description of `WSAdminPolicies` and its elements can be found in the ImportAdminRole section of this API chapter.

Other returnCodes and faultstrings

Return Code	Faultstring
-292	Admin is not authorized to access Administrator functions
-311	Must be a master admin to import admin policies; Assignable roles can only be imported by master admins
-312	Master admins can only be imported or modified by master admins

ImportAdminRole

Overview

The **ImportAdminRole** API enables the web service client to import administrator roles to IPAM.

Note that only administrators of administrator type "MASTER" can invoke this API.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **importAdminRole** request and response messages.

```
<wsdl:message name="importAdminRoleRequest">
  <wsdl:part name="wsadminRole" type="tns2:WSAdminRole"/>
</wsdl:message>
<wsdl:message name="importAdminRoleResponse">
  <wsdl:part name="importAdminRoleReturn" type="soapenc:string"/>
</wsdl:message>
```

Response

The string returned in the response contains the name of the administrator role allocated.

Request

The complex type **WSAdminRole**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSAdminRole

Below is the portion of *Imports.wsdl* that describes **WSAdminRole**, the parameter structure passed to importAdminRole.

```
<complexType name="WSAdminRole">
    <sequence>
     <element name="description" nillable="true" type="soapenc:string"/>
     <element name="id" nillable="true" type="soapenc:int"/>
     <element name="policies" nillable="true" type="tns2:WSAdminPolicies"/>
     <element name="roleName" nillable="true" type="soapenc:string"/>|
    </sequence>
</complexType>
```

Element	Description and accepted values	Required	Return Code	Faultstring
description	Specify a description of this administrator role.	no		
id	The internal ID of this administrator role, as provided by a GetAdminRole call. If this is set, the administrator role is updated instead of added.	yes, for modify	-303	Admin role ID= *id* not found

Element	Description and accepted values	Required	Return Code	Faultstring
policies	A WSAdminPolicies structure, described below.	no		
roleName	The name of the administrator role to be added or modified.	yes	-300 -301	Admininistrator role name required Duplicate adminstrator role name: *name*

`WSAdminRole` contains the parameter structure **`WSAdminPolicies`**:

```
<complexType name="WSAdminPolicies">
  <sequence>
    <element name="addressTypeAccess" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="authorizedFunctions" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element maxOccurs="unbounded" name="blockInfo" nillable="true" type="tns2:WSBlockACLInfo"/>
    <element maxOccurs="unbounded" name="blockTypeSizeAccess" nillable="true"
type="tns2:WSBlockTypeSizeACL"/>
    <element maxOccurs="unbounded" name="containerInfo" nillable="true" type="tns2:WSContainerACLInfo"/>
    <element name="deviceTypeAccess" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element maxOccurs="unbounded" name="domainAccess" nillable="true" type="tns2:WSDomainACL"/>
    <element maxOccurs="unbounded" name="netServiceAccess" nillable="true" type="tns2:WSNetServiceACL"/>
    <element name="policies" nillable="true" type="tns2:WSAdminOtherPolicies"/>
    <element name="resourceRecTypeAccess" nillable="true" type="impl:ArrayOf_soapenc_string"/>
  </sequence>
</complexType>
```

Element	Description and accepted values	Required	Return Code	Faultstring
addressTypeAccess	**ALL**, **NONE**, or a list of allowed address types. Only required if there are limitations. Accepted values are: **Static**, **Dynamic DHCP**, **Automatic DHCP**, **Manual DHCP, Reserved**, **Dynamic NA DHCPv6**, **Automatic NA DHCPv6**, **Manual NA DHCPv6**, **Dynamic TA DHCPv6, Automatic TA DHCPv6** and **Interface**. Defaults to **ALL**.	no	-73	Invalid address type: *type*
authorizedFunctions	**ALL**, **NONE**, or a list of functions and main headings the administrator can access. Each function corresponds to a menu item that this administrator is allowed to access. List can include a main heading like **IPAM**. List can include individual functions like **AddSites**. A complete list of keywords for the functions and headings is provided in the ImportAdminRole CLI section. These keywords are case-insensitive. Defaults to **ALL**.	no	-299	Invalid function: *function*
blockInfo	Limit the blocks the administrator can access. Specify a list of WSBlockACLInfo structures, described below.	no		

Element	Description and accepted values	Required	Return Code	Faultstring
blockTypeSizeAccess	Limit the block types and sizes an administrator can access. Specify a list of WSBlockTypeSizeACL structures, described below.	no		
containerInfo	Limit the containers the administrator can access. Specify a list of WSContainerACL structures, described below.	no		
deviceTypeAccess	**ALL**, **NONE**, or a list of allowed device types, as configured in IPAM. Only required if there are limitations. Defaults to **ALL**.	no	-52	Invalid device type: *type*
domainAccess	Limit the domains the administrator can access. Specify a list of WSDomainACL structures, described below.	no		
netServiceAccess	Limit the network services the administrator can access. Specify a list of WSNetServiceACL structures, described below.	no		
Policies	Specify the policies for this administrator. Specify a WSAdminOtherPolicies structure, as described below.	no		
resourceRecTypeAccess	**ALL**, **NONE**, or a list of allowed resource record types. Only required if there are limitations. Defaults to **ALL**. *Note: "NSAP-PTR" is specified as* ***NSAPPTR*** *and "Zone RR" is specified as* ***ZONERR***.	no	-92	Invalid resource record type: *type*

`WSAdminPolicies` contains the following parameter structures:

`WSBlockACLInfo`

A list of blocks with access indication. This is used to fine-tune the blocks given access by the container access control rules. You typically add blocks to an administrator role ACL when you want to override the privileges defined on the container for a specific block.

```
<complexType name="WSBlockACLInfo">
  <sequence>
   <element name="allowDelete" type="xsd:boolean"/>
   <element name="allowRead" type="xsd:boolean"/>
   <element name="allowWrite" type="xsd:boolean"/>
   <element name="blockName" nillable="true" type="soapenc:string"/>
   <element name="blockStatus" nillable="true" type="soapenc:string"/>
   <element name="containerName" nillable="true" type="soapenc:string"/>
   <element name="deviceApproveAccess" type="xsd:boolean"/>
  </sequence>
</complexType>
```

Element	Description and accepted values	Required	Return Code	Faultstring
allowDelete	Administrator may delete this block. Specify **true** or **false**.	no		
allowRead	Administrator may read this block. Specify **true** or **false**.	no		
allowWrite	Administrator may write to this block. Specify **true** or **false**.	no		
blockName	The name of the block for which access control is being defined. This defaults to reflect the access defined by the container access control rules. Specify **NONE** to indicate no block access rules (default) or to clear block access rules in overwrite mode. Note: For error -97, when multiple blocks with the same name are found in a container, specify blockStatus to resolve. This typically occurs when an aggregate block and another block have the same block name.	yes	-36 -97 -302	Block not found: *blockname* Multiple blocks with name: *blockname* found in container: *containrname* Must specify a container acl for this block acl: *block name* in container: *container name*
blockStatus	The status of the block for which access control is being defined. When multiple blocks with the same name are found in a container, specify blockStatus to resolve the conflict. This typically occurs when an aggregate block and another block have the same block name.	No	-17	Invalid block status: *status*
containerName	The name of the container containing the block in blockName.	yes	-5 -6 -7 -42	Container not found Invalid container name Ambiguous container name Container name is null
deviceApproveAccess	When workflow type system policy is set to "Device", indicates whether or not this administrator is an approver for changes to this block. Specify **true** or **false**.	no		

WSBlockTypeSizeACL

```
<complexType name="WSBlockTypeSizeACL">
  <sequence>
   <element name="blockType" nillable="true" type="soapenc:string"/>
   <element name="v4Abstain" type="xsd:boolean"/>
   <element name="v4BlockSize" nillable="true" type="impl:ArrayOf_soapenc_int"/>
   <element name="v6Abstain" type="xsd:boolean"/>
   <element name="v6BlockSize" nillable="true" type="impl:ArrayOf_soapenc_int"/>
  </sequence>
 </complexType>
```

Element	Description and accepted values	Required	Return Code	Faultstring
blockType	**ALL**, **NONE**, or a list of allowed block types. Only required if there are limitations. Defaults to **ALL**.	no	-13	Invalid block type: *type*
v4Abstain	Specify **true** or **false. true** indicates for IPV4 blocks: "Abstain and leave block sizes undecided".	no		
v4BlockSize	A list of IPV4 block sizes allowed for this block type. If no sizes are listed, the v4Abstain flag indicates whether all (true) or none (false) are allowed.	no	-194	Invalid v4 block size: *size*
v6Abstain	Specify **true** or **false. true** indicates for IPV6 blocks: "Abstain and leave block sizes undecided".	no		
V6BlockSize	A list of IPV6 block sizes allowed for this block type. If no sizes are listed, the v6Abstain flag indicates whether all (true) or none (false) are allowed.	no	-195	Invalid v6 block size: *size*

WSContainerACLInfo

```
<complexType name="WSContainerACLInfo">
  <sequence>
   <element name="allowDelete" type="xsd:boolean"/>
   <element name="allowRead" type="xsd:boolean"/>
   <element name="allowWrite" type="xsd:boolean"/>
   <element name="applyToChildren" type="xsd:boolean"/>
   <element name="containerName" nillable="true" type="soapenc:string"/>
   <element name="deviceApproveAccess" type="xsd:boolean"/>
  </sequence>
 </complexType>
```

Element	Description and accepted values	Required	Return Code	Faultstring
allowDelete	Administrator may delete from this container. Specify **true** or **false**.	no		
allowRead	Administrator may read from this container. Specify **true** or **false**.	no		
allowWrite	Administrator may write to this container. Specify **true** or **false**.	no		
applyToChildren	Determines whether the read, write, and delete priveleges also apply to the children of this container. Specify **true** or **false**.	no		

Element	Description and accepted values	Required	Return Code	Faultstring
containerName	The name of the container for which access control is being defined. Specify **NONE** to indicate no container access permitted. Specify **ALL** to indicate a rule with be set up for the root container permitting full access, including "apply to children". The default is **ALL** for ImportAdminRole, and **NONE** for ImportAdmin.	yes	-5 -6 -7 -42	Container not found Invalid container name Ambiguous container name Container name is null
deviceApproveAccess	When workflow type system policy is set to "Device", indicates whether or not this administrator is an approver for changes to this container. Specify **true** or **false**.	no		

WSDomainACL

```
<complexType name="WSDomainACL">
  <sequence>
   <element name="RRAccess" type="xsd:boolean"/>
   <element name="RRApproveAccess" type="xsd:boolean"/>
   <element name="RRWriteAccess" type="xsd:boolean"/>
   <element name="allowDelete" type="xsd:boolean"/>
   <element name="allowRead" type="xsd:boolean"/>
   <element name="allowWrite" type="xsd:boolean"/>
   <element name="applyToChildren" type="xsd:boolean"/>
   <element name="domainName" nillable="true" type="soapenc:string"/>
   <element name="domainType" nillable="true" type="soapenc:string"/>
  </sequence>
 </complexType>
```

Element	Description and accepted values	Required	Return Code	Faultstring
RRAccess	Administrator may read resource records within this domain. Specify **true** or **false**.	no		
RRApproveAccess	When Workflow Type system policy is set to "Resource Records", indicates whether or not administrator is a resource record approver for changes to this domain. Specify **true** or **false**.	no		
RRWriteAccess	Administrator may read resource records within this domain. Specify **true** or **false**.	no		
allowDelete	Administrator may delete this domain. Specify **true** or **false**.	no		
allowRead	Administrator may view attributes of this domain excluding resource records. This includes viewing attributes of zones for this domain. Specify **true** or **false**.	no		
allowWrite	Administrator may make changes to this domain, including changing SOA values, creating child domains and zones. Specify **true** or **false**.	no		

Element	Description and accepted values	Required	Return Code	Faultstring
applyToChildren	Determines whether these privileges also apply to the children of this domain. . Specify **true** or **false**.	no		
domainName	Specify **ALL**, **NONE** or the name of the domain for which access control is being defined. The default is **ALL** for ImportAdminRole, and **NONE** for ImportAdmin.	yes	-60	Domain not found: *name* / *type*
domainType	The domain type of the domain for which access control is being defined. Defaults to "Default".	no	-81	Domain type not found: *type*

WSNetServiceACL

```
<complexType name="WSNetServiceACL">
  <sequence>
   <element name="allowDeploy" type="xsd:boolean"/>
   <element name="allowRead" type="xsd:boolean"/>
   <element name="allowWrite" type="xsd:boolean"/>
   <element name="serverName" nillable="true" type="soapenc:string"/>
   <element name="serverType" nillable="true" type="soapenc:string"/>
  </sequence>
 </complexType>
```

Element	Description and accepted values	Required	Return Code	Faultstring
allowDeploy	Administrator may deploy this network service. Specify **true** or **false**.	no		
allowRead	Administrator may view attributes of this network service. Specify **true** or **false**.	no		
allowWrite	Administrator may change attributes of this network service. Specify **true** or **false**.	no		
serverName	The name of the network service for which access control is being defined.	yes	-186	Network Service Name is required
serverType	Specify the server type as either **DNS** or **DHCP**.	yes	-185	Network Service Name not found for name/type: *name* / *type*

WSAdminOtherPolicies.

```
<complexType name="WSAdminOtherPolicies">
  <sequence>
   <element name="allowCLIAccess" type="xsd:boolean"/>
   <element name="allowDupARecCheck" nillable="true" type="soapenc:string"/>
   <element name="allowDupHostnameCheck" nillable="true" type="soapenc:string"/>
   <element name="allowDupHwAddrCheck" nillable="true" type="soapenc:string"/>
   <element name="dupHostnameCheckingStyle" nillable="true" type="soapenc:string"/>
  </sequence>
 </complexType>
```

Element	Description and accepted values	Required	Return Code	Faultstring
allowCLIAccess	Administrator may use CLIs. Specify **true** or **false**. Defaults to **false**.	no		
allowDupARecCheck	Specify allow duplicate resource records for type A records as: **I**(gnore), **W**(arn) or **F**(ail). Defaults to **W**.	no	-298	Invalid dup A rec checking: *value*
allowDupHostnameCheck	Specify allow duplicate hostnames as: **I**(gnore), **W**(arn) or **F**(ail). Defaults to **W**.	no	-298	Invalid dup hostname checking: *value*
allowDupHwAddrCheck	Specify allow duplicate hardware addresses as: **I**(gnore), **W**(arn) or **F**(ail). Defaults to **W**.	no	-298	Invalid dup hardware address checking: *value*
dupHostnameCheckingStyle	Specify the duplicate hostname checking style as: **F** (fully qualified) or **H** (hostname only). Default is **F**.	no	-298	Invalid dup hostname checking style: *value*

Other returnCodes and faultstrings

Return Code	Faultstring
-292	Admin is not authorized to access Administrator functions
-313	Admin must be MASTER for import admin role
-314	Invalid to modify a hidden role. Use ImportAdmin with overwrite.

ImportAggregateBlock

Overview

The **ImportAggregateBlock** API enables the web service client to insert an intermediate level aggregate block between existing blocks in the block hierarchy. By specifying a parent block, target block, and a container, the service will handle validating and inserting the desired aggregate block. The service will also adjust the parent block assignments of any would-be child blocks.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **importAggregateBlock** request and response messages.

```
<wsdl:message name="importAggregateBlockRequest">
                          <wsdl:part name="aggregateBlock" type="tns2:WSAggregateBlock"/>
</wsdl:message>
<wsdl:message name="importAggregateBlockResponse">
```

Response

There is no data in the response.

Request

The complex type **WSAggregateBlock**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSAggregateBlock

Below is the portion of *Imports.wsdl* that describes **WSAggregateBlock**, the parameter structure passed to **importAggregateBlock**. The elements are described in the table that follows.

```
<complexType name="WSAggregateBlock">
  <sequence>
    <element name="SWIPname" nillable="true" type="soapenc:string"/>
    <element name="allocationReason" nillable="true" type="soapenc:string"/>
    <element name="allocationReasonDescription" nillable="true" type="soapenc:string"/>
    <element name="blockAddr" nillable="true" type="soapenc:string"/>
    <element name="blockName" nillable="true" type="soapenc:string"/>
    <element name="blockSize" nillable="true" type="soapenc:int"/>
    <element name="blockType" nillable="true" type="soapenc:string"/>
    <element name="container" nillable="true" type="soapenc:string"/>
    <element name="createReverseDomains" type="xsd:boolean"/>
    <element name="description" nillable="true" type="soapenc:string"/>
    <element name="domainType" nillable="true" type="soapenc:string"/>
    <element name="interfaceAddress" nillable="true" type="soapenc:string"/>
    <element name="interfaceName" nillable="true" type="soapenc:string"/>
    <element name="parentBlockAddr" nillable="true" type="soapenc:string"/>
    <element name="parentBlockContainer" nillable="true" type="soapenc:string"/>
    <element name="parentBlockSize" nillable="true" type="soapenc:int"/>
    <element name="userDefinedFields" nillable="true" type="impl:ArrayOf_soapenc_string"/>
```

```
  </sequence>
</complexType>
```

Element	Description and accepted values	Required	Return Code	Faultstring
container	The name of the container into which to insert the new aggregate block. Names can be in either short or long format. Short format example: Dallas. Long format example: InControl/Texas/Dallas. Long format eliminates ambiguity in cases where there are duplicate container names.	Yes	-42 -5 -99	Missing container name Could not find container: <container> Admin is not authorized to add blocks to this container
blockAddr	The start address of the new aggregate block.	Yes	-127 -12	Block start and parent block address both required Could not convert <address> to ipAddress
blockSize	The size of the block in short-notation (e.g., 24 for a 255.255.255.0 network).	Yes	-15	Block size invalid: <size>
blockType	The Block Type for the block If not specified, a block type of Any is assumed.	No	-13	Invalid block type <type>
blockName	A name for the block. Defaults to system supplied name of Address space/Block size.	No		
description	A description of the block.	No		
SWIPname	SWIP name for the block.	Yes, if required by container rules	-66 -70	SWIPname is required for this container/blocktype SWIPname is not allowed for this container/blocktype
Allocation Reason	The name of a pre-existing Allocation Reason.	No	-22	Invalid allocation reason: <reason>
Allocation Reason Description	A description of the reason for the allocation. Wrap the statement in "quotes" if it contains any commas.	No		
Interface Name	If this block is being added to a device container, the name of the interface to attach the block to.	Yes, if block is being added to device container. Otherwise, no.	-20 -19 -5	Missing interface name No interface found Could not find containerID=<id>
Interface Address	DEPRECATED. This field will be ignored.	No		

Element	Description and accepted values	Required	Return Code	Faultstring
Create Reverse Domains	Whether or not to automatically create reverse DNS domain(s) for this block. Accepted values are true or false. If not specified, defaults to false.	No	-82	createReverseDomains must be true or false: *value*
domainType	Specify the domain type for the reverse DNS domain(s) to be created when Create Reverse Domains is "true". If not specified, defaults to "Default".	No	-2 -81	Exception retrieving domain type: <type> Domain type not found: <type>
userDefined Fields	A series of name=value pairs, where the name is the UDF name and the value is desired value. Multiple pairs can be specified by separating each pair with the "|" character. For example, UDFone=value one |UDFtwo=value two. If the UDF type is Checkbox, the valid values are "on" or "off".	Yes, for UDFs defined as required fields.	-53 -63 -64	SQL Exception retrieving valid UDF list Invalid UDF list or button selection: <value> *or* Invalid UDF: <name> *or* There are no UDFs defined for this container and block type Missing required UDF: <name>
parentBlock Container	The name of the container where the parent block resides.	Yes	-42 -5 -99	Missing container name Could not find container: <container> Admin is not authorized to add blocks to this container
parentBlock Addr	The address of the parent block	Yes	-127 -12	Block start and parent block address both required Could not convert <address> to ipAddress
parentBlock Size	The size of the parent block in short-notation (e.g., 24 for a 255.255.255.0 network).	Yes	-15	Block size invalid: <size>

ImportChildBlock

Overview

The **ImportChildBlock** API enables the web service client to import child blocks into IPAM. This API is used to define sub-allocations of address space, taken from parent address space. This space is allocated from the parent, and then marked with the status that is specified in the request. The name of the block allocated is returned to the client application in the response.

This API can also be used to attach existing blocks to another container by specifying an existing blockAddr.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **importChildBlock** request and response messages.

```
<wsdl:message name="importChildBlockResponse">
  <wsdl:part name="importChildBlockReturn" type="soapenc:string" />
</wsdl:message>
<wsdl:message name="importChildBlockRequest">
  <wsdl:part name="inpChildBlock" type="tns1:WSChildBlock" />
  <wsdl:part name="inpBlockPolicy" type="tns1:WSSubnetPolicy" />
</wsdl:message>
```

Response

The string returned in the response contains the name of the block allocated, for example, `10.0.0.128/28`.

Request

The complex types **WSChildBlock** and **WSSubnetPolicy**, which are passed as input from the client to the web service, are described in the next section.

Parameters

WSChildBlock

Below is the portion of *Imports.wsdl* that describes **WSChildBlock**, the first parameter structure passed to **importChildBlock**. The elements are described in the table that follows.

```
<complexType name="WSChildBlock">
  <sequence>
    <element name="SWIPname" nillable="true" type="xsd:string" />
    <element name="allocationReason" nillable="true" type="xsd:string" />
    <element name="allocationReasonDescription" nillable="true" type="xsd:string" />
    <element name="allocationTemplate" nillable="true" type="xsd:string" />
    <element name="blockAddr" nillable="true" type="xsd:string" />
    <element name="blockName" nillable="true" type="xsd:string" />
    <element name="blockSize" nillable="true" type="xsd:string" />
    <element name="blockStatus" nillable="true" type="xsd:string" />
    <element name="blockType" nillable="true" type="xsd:string" />
```

```
    <element name="container" nillable="true" type="xsd:string" />
    <element name="createReverseDomains" nillable="true" type="xsd:string" />
    <element name="description" nillable="true" type="xsd:string" />
    <element name="domainType" nillable="true" type="xsd:string" />
    <element name="interfaceAddress" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="interfaceName" nillable="true" type="xsd:string" />
    <element name="ipv6" type="xsd:boolean"/>
    <element name="primarySubnet" type="xsd:boolean"/>
    <element name="nonBroadcast" type="xsd:boolean"/>
    <element name="userDefinedFields" nillable="true" type="impl:ArrayOf_xsd_string" />
    <element name="excludeFromDiscovery" nillable="true" type="xsd:string" />
  </sequence>
</complexType>
```

Element	Description and accepted values	Required	Return Code	Faultstring
SWIPname	SWIP name for this block	yes, for containers with rules requiring it	-66 -70	SWIP name required for this block SWIP name not allowed for this block
allocationReason	The name of a pre-existing Allocation Reason. If Allocation Reason is not currently in IPAM, this field is skipped.	no	-22	Invalid reason code
allocationReason Description	A description of the reason for the allocation.	no		No validation required
allocationTemplate	If this block is being added to a device container with blockStatus=**Deployed**, the name of the allocation template to use to create address pools from the newly created block.	no	-65 -69 -71	Invalid allocation template: *template* Allocation template offsets invalid for this block Address pool creation failed.
blockAddr	The address block to allocate. If no address block is specified, space will be auto-allocated. If the address is specified and already exists, the block will be attached to the specified container.	no	-12	Invalid block address: *blockAddr*
blockName	A name for the block. Defaults to system supplied name of *Address space/Block size*.	no		
blockSize	The size of the block in short-notation (e.g., 24 for a 255.255.255.0 network).	yes	-15	Invalid block size: *blockSize*
blockStatus	The current status of the block. Accepted values are: **Deployed**, **FullyAssigned**, **Reserved**, **Aggregate**.	yes	-17	Invalid block status: *blockStatus*
blockType	The Block Type for the block If not specified, a block type of **Any** is assumed.	no	-13	Invalid block type: *blockType*

Element	Description and accepted values	Required	Return Code	Faultstring
container	The name of the container that will hold the block. Names can be in either short or long format. Short format example: **Dallas**. Long format example: **InControl/Texas/Dallas**. Long format eliminates ambiguity in cases where there are duplicate container names.	yes	-5 -6 -7 -1 -8	Container not found: *containerName* Invalid container name: *containerName* Container name ambiguous: *containerName* Database error Could not attach block to this container.
createReverseDomains	Whether or not to automatically create reverse DNS domain(s) for this block. Accepted values are **true** or **false**. If not specified, defaults to **false**.	no	-82	createReverseDomains must be true or false: *value*
Description	A description of the block. Use "\n" to separate lines.	no	-198	Block Description cannot exceed 255 characters.
domainType	Specify the domain type for the reverse DNS domain(s) to be created when Create Reverse Domains is "true". If not specified, defaults to "Default".	no	-81	Domain type not found: *domainType*
interfaceAddress	The specific addresses, or offsets from the beginning, for the interface IP address(es). If an IP address is specified, it should be in the form xxx.xxx.xxx.xxx. If an integer is specified, it will be interpreted as an offset from the beginning of the block (i.e. an offset of 2 in a /24 block will create an interface xxx.xxx.xxx.2). This can also be the string "from-start" or "from-end" if you are attaching a block to a device container and wish IPAM to determine the first available address from the start or the end of the block. An offset of 1 is assumed if none is specified.	no	-18 -21	Invalid interface address: *interfaceAddress* Invalid interface offset: *interfaceAddress*

Element	Description and accepted values	Required	Return Code	Faultstring
interfaceName	If this block is being added to a device container, the name of the interface to attach the block to.	yes, for device containers only	-20 -19 -24	Missing interface name Invalid interface name: *interfaceName* No interface name specified, could not attach to device container.
ipv6	True if this is an IPV6 block. If not specified, defaults to **false**.	No		
primarySubnet	Flag indicating if this subnet is primary. Accepted values are **true** or **false**. If not specified, defaults to **false**. **Valid only for Deployed blocks.**	No	-17	importBlock set primary subnet failed: flag supported only for In-Use/Deployed blocks
nonBroadcast	Flag indicating if this is a Non-Broadcast subnet. Non-broadcast subnets are allowed to assign devices to the subnet and broadcast addresses. Accepted values are **true** or **false**. If not specified, defaults to **false.** **Valid for IPv4 Deployed blocks only.**	No	-17	importBlock set nonBroadcast failed: flag supported only for In-Use/Deployed blocks
userDefinedFields	A string array containing one or more *name=value* pairs, where the *name* is the UDF name and the *value* is the desired value, for example, **State=PA.**	yes, for UDFs defined as required fields	-63 -64	Invalid UDF: *udf* Missing required UDF: *udf*
excludeFromDiscovery	Whether or not to exclude this subnet from Host Discovery tasks. Accepted values are **true** or **false**. If not specified, defaults to **false**.	no	-82 -13	excludeFromDiscovery must be true or false: *value* Invalid Block Type: flag supported for Deployed blocks (Subnets)

WSSubnetPolicy

Below is the portion of *Imports.wsdl* that describes **`WSSubnetPolicy`**, the second parameter structure passed to **`importChildBlock`**. The elements are described in the table that follows.

```
 <complexType name="WSSubnetPolicy">
<sequence>
  <element name="DHCPOptionsSet" nillable="true" type="soapenc:string" />
  <element name="DHCPPolicySet" nillable="true" type="soapenc:string" />
  <element name="DNSServers" nillable="true" type="impl:ArrayOf_soapenc_string" />
```

```
    <element name="cascadePrimaryDhcpServer" nillable="true" type="xsd:boolean" />
    <element name="defaultGateway" nillable="true" type="soapenc:string" />
    <element name="failoverDHCPServer" nillable="true" type="soapenc:string" />
    <element name="forwardDomainTypes" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="forwardDomains" nillable="true" type="impl:ArrayOf_soapenc_string" />
    <element name="networkLink" nillable="true" type="soapenc:string"/>
    <element name="primaryDHCPServer" nillable="true" type="soapenc:string" />
    <element name="primaryWINSServer" nillable="true" type="soapenc:string" />
    <element name="reverseDomainTypes" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="reverseDomains" nillable="true" type="impl:ArrayOf_soapenc_string" />
  </sequence>
</complexType>
```

Element	Description and accepted values	Required	Return Code	Faultstring
DHCPOptionsSet	The name of a previously defined DHCP Options set.	no	-67	DHCP Option set *set* not found
DHCPPolicySet	The name of a previously defined DHCP policy set.	no	-68	DHCP Policy set *set* not found
DNSServers	The name of previously defined DNS servers to be sent as an IP address to the client.	no	-93	DNS Server *server* not found
cascadePrimary DHCPServer	If address pool or individual IP address objects within the subnet reference a specific DHCP server, this attribute can be used to allow the 'primaryDHCPServer' attribute value to "cascade" to these address pool and IP address objects. Note that address pool and IP address objects that are configured with "Same as Subnet" for the primary DHCP server will be unaffected. Used for overwrite only.	no		
defaultGateway	The default gateway that DHCP clients on this subnet will use. Accepted value is an IP address.	no	-26	Invalid IP Address: *ipaddress*
failover DHCPServer	The name of the DHCP server that will act as failover for this subnet. This cannot be the same as the primary DHCP server.	no	-94	DHCP Server *server* not found
Forward Domains	The forward domain names that will available to the user when adding an IP address to the system. The first forward domain in the list will be used when there is a domain name DHCP option.	no	-95	Invalid forward DNS Domain *domain*
forwardDomain Types	The domain types corresponding to the domains listed in forwardDomains. Only required for non-default domain types.	no	-81	Domain type not found: *domainType*
networkLink	**Valid for deployed blocks in logical containers only**, the name of a logical network link already defined in IPAM. This is the name of the Shared Network Segment for this subnet. This value is automatically assigned for blocks in device containers.	no	-279 -280	Network Link not found, name=*name* Network link valid only for logical containers
primaryDHCP Server	The name of the DHCP server that will act as primary for this subnet.	no	-94	DHCP Server *server* not found

Element	Description and accepted values	Required	Return Code	Faultstring
primaryWINS Server	The IP address of the Microsoft WINS server for clients in this subnet.	no	-26	Invalid IP Address: *ipaddress*
reverseDomains	The reverse domain names that will available to the user when adding an IP address to the system.	no	-95	Invalid reverse DNS Domain *domain*
reverseDomain Types	The domain types corresponding to the domains listed in reverseDomains. Only required for non-default domain types.	no	-81	Domain type not found: *domainType*

Other returnCodes and faultstrings

Return Code	Faultstring
-2	Allocation failed
-3	Invalid arguments *missing xxx parameter*
-8	Exception attaching block: *msg*
-9	Subnet policy record creation failed
-23	Candidate block not found
-53	SQL Exception
-96	Failover DHCP specified without primary
-96	Failover must be different than Primary
-99	Access Denied. The administrator does not have rights to add any blocks OR the administrator does not have rights to add blocks to the specified container.

ImportContainer

Overview

The **ImportContainer** API enables the web service client to import containers into IPAM. These can be logical containers or device containers. It can also be used to modify existing containers

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **importContainer** request and response messages.

```
<wsdl:message name="importContainerResponse" />
<wsdl:message name="importContainerRequest">
  <wsdl:part name="inpContainer" type="tns1:WSContainer" />
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSContainer**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSContainer

Below is the portion of *Imports.wsdl* that describes **WSContainer**, the parameter structure passed to **importContainer**. The elements are described in the table that follows.

```
<complexType name="WSContainer">
  <sequence>
    <element name="allowedAllocFromParentBlocktypes" nillable="true"
             type="impl:ArrayOf soapenc string"/>
    <element name="allowedBlockTypes" nillable="true"
             type="impl:ArrayOf_soapenc_string"/>
    <element name="allowedDeviceTypes" nillable="true"
             type="impl:ArrayOf soapenc string"/>
    <element name="allowedRootBlockTypes" nillable="true"
             type="impl:ArrayOf_soapenc_string"/>
    <element name="applyDHCPToMultiparentDevContainer"
             type="xsd:boolean"/>
    <element name="blockTypeInfoTemplates" nillable="true"
             type="impl:ArrayOf soapenc string"/>
    <element name="containerName" nillable="true"
             type="soapenc:string"/>
    <element name="containerType" nillable="true"
             type="soapenc:string"/>
    <element name="description" nillable="true"
             type="soapenc:string"/>
    <element name="deviceInfoTemplates" nillable="true"
             type="impl:ArrayOf soapenc_string"/>
    <element name="id" nillable="true"
             type="soapenc:int"/>
    <element name="ignoreBlocktypeInUse"
             type="xsd:boolean"/>
```

```
    <element name="informationTemplate" nillable="true"
             type="impl:ArrayOf_soapenc_string"/>
    <element name="maintainHistoryRecs"
             type="xsd:boolean"/>
    <element name="parentName" nillable="true"
             type="soapenc:string"/>
   <element name="replaceDHCPServer" nillable="true"
             type="soapenc:string"/>
   <element name="replaceDHCPServerV6" nillable="true"
             type="soapenc:string"/>
    <element name="requireSWIPNameBlockTypes" nillable="true"
             type="impl:ArrayOf soapenc string"/>
    <element name="userDefinedFields" nillable="true"
             type="impl:ArrayOf_soapenc_string"/>
  </sequence>
</complexType>
```

Element	Description and accepted values	Required	Return Code	Faultstring
allowedAllocFrom ParentBlocktypes	A string array containing a listing of the block types enabled for Rule 3: "Allow space allocation from Parent container for block type".	no	-13	Invalid blocktype rule 3: *blocktype*
allowedBlock Types	A string array containing a listing of the block types enabled for Rule 1: "Container may contain blocks of type".	no	-13	Invalid blocktype rule 1: *blocktype*
allowedDevice Types	A string array containing a listing of the device types enabled for Rule 5: "Container may contain devices of type". To specify that all device types should be allowed, use **ALL** as the first element in the array. To specify that no device types should be allowed, use **NONE**. **ALL** is the default.	no	-58	Invalid device type: *devicetype*
allowedRootBlock Types	A string array containing a listing of the block types enabled for Rule 2: "Allow Root Blocks to be added to this container of block type".	no	-13	Invalid blocktype rule 2: *blocktype*
applyDHCPTo MultiparentDev Container	A boolean used only in conjunction with the 'replaceDhcpServer' attribute, when replacing DHCP servers in objects during a container move operation. This flag indicates if those changes should affect objects attached to multiparented device containers, which may have differing DHCP server associations along each container anscestory. Note: Multiparented device containers cannot be moved using API/CLIs. This attribute applies when moving a logical container with a multi-parented device container along the logical container ancestory.	no		

Element	Description and accepted values	Required	Return Code	Faultstring
blockTypeInfo Templates	A string array containing the list of information templates to be associated with each of the blocks allocated to this container according to their blocktype. The order must match the blocktypes listed in the allowedBlockTypes parameter, and the length of this array is the same as that parameter. If a blocktype does not have a template associated with it, set that array element to null or blank.	no	-3 -78	Error occurred while updating device info templates for container Invalid Information Template: *templateName*
containerName	The name of the container. If you are creating a device container, this container name must match exactly the name of a network element already in the database or the request will be rejected.	yes	-3 -4 -9 -2 -132 -143 -158	Invalid arguments: container name, parent container name, or container type is null Duplicate container name: *containerName* Invalid device container: *containerName* Exception finding network element: *exception message* Invalid argument the name provided is ambiguous the new parent has a container with the same name
containerType	Either logical or device.	yes	-3 -8	Invalid arguments: container name, parent container name, or container type is null Invalid container type: *type*
description	A brief description of the container. Use "\n" to separate lines.	no	-198	Container Description cannot exceed 255 characters
deviceInfo Templates	A string array containing the list of information templates to be associated with each of the devices allocated to this container according to their device type. The order must match the device types listed in the allowedDeviceTypes parameter, and the length of this array is the same as that parameter. If a device type does not have a template associated with it, set that array element to null or blank.	no	-3 -78	Error occurred while updating blocktype info templates for container Invalid Information Template: *templateName*
id	The internal ID of this container, as provided by the getContainerByName call. If this is set, the container is updated instead of added.	Yes, for modify only	-5	Could not find container for modify: *id*

Element	Description and accepted values	Required	Return Code	Faultstring
ignoreBlocktypeInUse	Set this to "true" when disallowing a block type in use by the container, indicated by the list in the allowedBlockTypes field.	no	-43	Blocktype in use by container: *container*
Information Template	A string array containing the list of information templates to be associated with this container.	no	-1 -78	DB error retrieving information template Invalid Information Template: *templateName*
maintainHistoryRecs	Specify whether or not Container History and Block History records will be kept for all appropriate block types. The history records are created each time the Global Utilization Rollup task is run. Accepted values are **true** or **false**. If not specified, defaults to **false**.	no		
parentName	The name of the parent container for this container. Names can be in either short or long format. Short format example: **Dallas**. Long format example: **InControl/Texas/Dallas**. Long format eliminates ambiguity in cases where there are duplicate container names.	yes	-3 -10 -11 -1 -132 -143	Invalid arguments: container name, parent container name, or container type is null Parent container not found: *containerName* Parent container name is ambiguous: *containerName* Database error Invalid argument the name provided is ambiguous
replaceDHCPServer	This attribute is used only when utilizing the IPAM feature of attaching DHCP servers to Containers, and when the modify operation involves moving a container. In this situation, Subnet, Address Pool, and IP Address objects that are moved as a result of the container move may become "invalid" with respect to the DHCP servers attached to the target container or target container's nearest anscestor. In order to update these objects' associated DHCP servers as part of the container move operation, specify the name of a DHCP server which is "valid" for the target container. This field will update any dynamic V4 objects and must be a V4 capable DHCP server.	no	-94 -13	DHCP Server not valid for this container or the user does not have write access The DHCP server was not found

Element	Description and accepted values	Required	Return Code	Faultstring
replaceDHCPServerV6	This is similar to the above "replaceDHCPServer" field, but will update any dynamic V6 objects and must be a V6 capable DHCP server.	no	-94 -13	DHCP Server not valid for this container or the user does not have write access The DHCP server was not found
requireSWIPNameBlockTypes	A string array containing a listing of the block types enabled for Rule 4: "Require SWIP Names on blocks of type".	no	-13	Invalid blocktype rule 4: *blocktype*
userDefinedFields	The user defined fields associated with this container, as listed in the container information templates specified in parameter informationTemplate. Specify as a string array containing one or more *name*=*value* pairs, where the *name* is the UDF name and the *value* is the desired value, for example, **State=PA.**	yes, for UDFs defined as required fields	-63 -64	Invalid UDF: *udf* Missing required UDF: *udf*

Other returnCodes and faultstrings

Return Code	Faultstring
-1	various unexpected DB errors
-2	Allocation failed
-3	Invalid arguments missing xxx parameter
-41	Container move failed. (Root container not moveable; the new parent is a descendant of the container; Move failed.)
-99	access denied

ImportDevice

Overview

The **ImportDevice** API enables the web service client to import devices into IPAM.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **importDevice** request and response messages.

```
<wsdl:message name="importDeviceResponse" />
<wsdl:message name="importDeviceRequest">
  <wsdl:part name="inpDevice" type="tns2:WSDevice" />
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSDevice**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSDevice

Below is the portion of *Imports.wsdl* that describes **WSDevice**, the parameter structure passed to importDevice. The elements are described in the table that follows.

```
<complexType name="WSDevice">
  <sequence>
    <element name="MACAddress" nillable="true" type="soapenc:string"/>
    <element name="addressType" nillable="true" type="soapenc:string"/>
    <element name="aliases" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="container" nillable="true" type="soapenc:string"/>
    <element name="description" nillable="true" type="soapenc:string"/>
    <element name="deviceType" nillable="true" type="soapenc:string"/>
    <element name="domainName" nillable="true" type="soapenc:string"/>
    <element name="domainType" nillable="true" type="soapenc:string"/>
    <element name="duid" nillable="true" type="soapenc:string"/>
    <element name="dupWarning" nillable="true" type="soapenc:string"/>
    <element name="hostname" nillable="true" type="soapenc:string"/>
    <element name="hwType" nillable="true" type="soapenc:string"/>
    <element name="ipAddress" nillable="true" type="soapenc:string"/>
    <element name="resourceRecordFlag" nillable="true" type="soapenc:string"/>
    <element name="userDefinedFields" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element maxOccurs="unbounded" name="interfaces" nillable="true" type="tns2:WSInterface"/>
    <element name="id" nillable="true" type="soapenc:int"/>
    <element name="excludeFromDiscovery" nillable="true" type="soapenc:string"/
  </sequence>
</complexType>

<complexType name="WSInterface">
  <sequence>
    <element name="addressType" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="container" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="excludeFromDiscovery" nillable="true" type="soapenc:string"/>
```

```
    <element name="hwType" nillable="true" type="soapenc:string"/>
    <element name="id" nillable="true" type="soapenc:int"/>
    <element name="ipAddress" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="macAddress" nillable="true" type="soapenc:string"/>
    <element name="name" nillable="true" type="soapenc:string"/>
    <element name="relayAgentCircuitId" nillable="true"
type="impl:ArrayOf_soapenc_string"/>
     <element name="relayAgentRemoteId" nillable="true"
type="impl:ArrayOf_soapenc_string"/>
    <element name="sequence" nillable="true" type="soapenc:int"/>
    <element name="virtual" nillable="true" type="impl:ArrayOf_soapenc_boolean"/>
</sequence>
</complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
MACAddress	The hardware MAC address of the device.	Yes, if hwtype is specified or if address type is Manual DHCP.	-77	MAC Address must be specified when using Hardware Type *or* Hardware address must be specified for address type Manual DHCP
addressType	The address type of this device. Accepted values are: **Static**, **Dynamic DHCP**, **Automatic DHCP**, **Manual DHCP** and **Reserved**. For DHCP V6: **Dynamic NA DHCPv6**, **Automatic NA DHCPv6**, **Manual NA DHCPv6**, **Dynamic TA DHCPv6** and **Automatic TA DHCPv6**. Note that if **Dynamic**, **Automatic** or **Manual DHCP** is specified, there must be a DHCP server defined in the subnet policies for this IP Address. Devices of type **Interface** may not be imported, but they can be modified using overwrite. However, the addressType and ipAddress cannot be updated. Use ImportChildBlock to modify or delete **Interface** IP addresses.	Yes	-73 -260 -261	Invalid address type: *type* Invalid Address Type for *IPv4 | IPv6*: *type* Cannot change the ipaddress of an interface Cannot change the address type to/from 'Interface'
Aliases	A string array containing the alias or list of aliases for this hostname. When you specify an alias, a CNAME record is created. The alias may be fully qualified (contains a trailing dot), or not. When fully qualified, everything that is after the first qualifier is interpreted as a domain name. When not fully qualified, the CNAME record will be created in the same domain as the device. To use this element, you must also specify resourceRecordFlag as **true**.	No		

Element	Accepted Values	Required	Return Code	Faultstring
container	The name of the container that contains the device. Use the container element of the interfaces type to resolve ambiguity for multi-homed devices when different containers are required.	Yes, if overlapping space is in use and the block name is ambiguous.	-2 -6 -7 -5 -1	Could not find container: *container* Invalid container name Ambiguous container name Container not found Database error
description	A description of the device. Use "\n" to separate lines.	No		
deviceType	The name of a device type configured in IPAM.	Yes, if hostname specifies use of naming policy.	-47, -58 -75	Device type not found: *type* Device type required when using naming policy
domainName	Domain name already defined to IPAM	Yes, if resource Record Flag is "true" and the block policy has no forward domains.	-60 -71	Domain not found: *domain* Domain required when adding resource records
domainType	Domain type already defined to IPAM. If not specified, the "Default" domain type will be used.	No	-59	Domain type not found: *view*
duid	DHCP Unique Identifier	Yes, for Address Type "Manual NA DHCPV6".	-259	DUID required for Manual NA DHCPV6
dupWarning	If the administrator policy of the user indicates "Warn" for the "Allow Duplicate Hostnames Checking" option, the warning will be ignored and the device added with the duplicate hostname when this field is **true**. Accepted values are **true** or **false**. If not specified, defaults to **false**.	No	-80	Duplicate hostname for hostname
hostname	Valid host name or **APPLYNAMINGPOLICY**.	Yes		

Element	Accepted Values	Required	Return Code	Faultstring
hwType	Specify **Ethernet** or **Token Ring**. When hwtype is specified, MACAddress must also be specified. If MACAddress is specified, this will default to **Ethernet**.	No	-72	Invalid Hardware type: *type*
ipAddress	The IP Addresses of the Device. To create multi-homed devices, use the interfaces element. To modify devices with multiple IP Addresses on a single interface use the interfaces element. To indicate that IPAM should use the next available address in a block, specify the block address, followed by "/from-start" or "/from-end", for example: 10.30.0.0/from-start 3FEE::/from-start	Yes	-26 -226 -227 -228	Invalid IP Address: *address* Invalid direction IPV6 not supported Block out of space
resourceRecordFlag	Whether or not to add resource records for this device. If not specified as **true**, defaults to false.	No		
userDefinedFields	A string array containing one or more *name=value* pairs, where the *name* is the UDF name and the *value* is the desired value, for example**, State=PA**. If the UDF type is Checkbox, the valid values are **on** and **off**.	Yes, for UDFs defined as required fields.	-53 -63 -64	store UDFs failed: SQL exception message Invalid UDF Missing required UDF: *udf*

Element	Accepted Values	Required	Return Code	Faultstring
interfaces	An array of WSInterface structures. Each element in the array corresponds to one interface for a multi-homed device or to the only interface for a single-homed device. The fields in the WSInterface structure are described below. Their accepted values are the same as in the WSDevice structure, unless otherwise noted: - addressType - container - excludeFromDiscovery - hwType - id: The internal ID provided by a GetDeviceBy call. Required for modify requests. - ipAddress - macAddress - name: Interface Name (Required) - relayAgentCircuitId: Ignored (populated by DHCPCollections task) - relayAgentRemoteId: Ignored (populated by DHCPCollections task) - sequence: Provided by GetDeviceBy call; not used - virtual: boolean indicating whether this interface address is virtual (See Guide to Using IPAM, "Configuring Shared and Virtual IP Addresses" for more information.)	Yes, for mulit-homed devices Yes, for all modify requests.	-18 -19 -20 -260 -261	Interface required when modifying a device IP Address missing or invalid Interface name missing. Cannot change the ipaddress of an interface Cannot change the address type to/from 'Interface'
id	The internal ID of this device, as provided by a GetDeviceBy call. If this is set, the device is updated instead of added.	Yes, for modify		
excludeFrom Discovery	Flag indicating if this device should be included in Host Discovery tasks. Accepted values are **true** or **false**. If not specified, defaults to **false**. For multihomed devices, the flag must be specified for each IP/Interface via the WSInterface structure.	No		

Other returnCodes and faultstrings

Return Code	Faultstring
-2	Import device failed (allocation failure)
-36	More than one block for address: *address* Specify Container
-62	Subnet not found for: *ip address*
-47	System error
-79	CNAME record could not be created

ImportDeviceResourceRecord

Overview

The **ImportDeviceResourceRecord** API enables the web service client to import DNS resource records for a device into IPAM.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **importDeviceResourceRecord** request and response messages.

```
<wsdl:message name=" importDeviceResourceRecordResponse" />
<wsdl:message name="importDeviceResourceRecordRequest">
  < <wsdl:part name="inpRR" type="tns2:WSDeviceResourceRec"/>
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSDeviceResourceRec**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSDeviceResourceRec

Below is the portion of *Imports.wsdl* that describes **WSDeviceResourceRec**, the parameter structure passed to **importDeviceResourceRecord**. The elements are described in the table that follows.

```
<complexType name="WSDeviceResourceRecord">
  <sequence>
    <element name="TTL" nillable="true" type="soapenc:string"/>
    <element name="comment" nillable="true" type="soapenc:string"/>
    <element name="container" nillable="true" type="soapenc:string"/>
    <element name="data" nillable="true" type="soapenc:string"/>
    <element name="domain" nillable="true" type="soapenc:string"/>
    <element name="domainType" nillable="true" type="soapenc:string"/>
    <element name="hostname" nillable="true" type="soapenc:string"/>
    <element name="id" nillable="true" type="soapenc:int"/>
    <element name="ipAddress" nillable="true" type="soapenc:string"/>
    <element name="owner" nillable="true" type="soapenc:string"/>
    <element name="resourceRecClass" nillable="true" type="soapenc:string"/>
    <element name="resourceRecType" nillable="true" type="soapenc:string"/>
  </sequence>
</complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
container	The name of the container that holds the device. This is required only if there is overlapping address space in use and the ip address is in overlapping space. The container is then used to uniquely determine the device.	No		
domainType	Domain type already defined to IPAM. If not specified, the "Default" domain type will be used.	No	-108	Domain not found: domainType/domain
domain	Domain name where resource records are to be added.	Yes.	-108 -135 N/A	Domain not found: domainType/domain Domain required Not authorized
hostname	The device host name.	Yes, unless IP Address is specified.	-121 -120	Hostname not unique: hostname No device with hostname:
ipAddress	The IP Address of the Device.	Yes, unless Host Name is specified.	-28 -120	IP Address not unique: ipAddress No device with ipAddress
owner	The owner field of the resource record.	Yes	-89 -106	Owner not specified Invalid character in owner
resourceRecClass	The Class of the Resource Record. Defaults to "IN".	No		
resourceRecord Type	The Type of the resource Record.	Yes	-92	Invalid Type
TTL	The Time To Live for the record.	No		
data	The data portion of the resource record. The format is dependent on the type specified above.	Yes	-91 -337	Data Required CNAME rdata must be canonical domain name
comment	Comment text associated with the resource record.	No		

ImportDhcpServer

Overview

The **ImportDhcpServer** API enables the web service client to create or DHCP servers in IPAM.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **importNetService** request and response messages.

```
<wsdl:message name="importDhcpServerResponse" />
<wsdl:message name="importDhcpServerRequest">
  <wsdl:part name="dhcpServer" type="tns2:WSDhcpServer" />
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSDhcpServer**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSDhcpServer

Below is the portion of Imports.wsdl that describes **WSDhcpServer**, the parameter structure passed to **importDhcpServer**. The elements are described in the table that follows.

```
<complexType name="WSDhcpServer">
  <sequence>
    <element name="agent" nillable="true" type="soapenc:string"/>
    <element name="beginExtension" nillable="true" type="soapenc:string"/>
    <element name="cliArgs" nillable="true" type="soapenc:string"/>
    <element name="cliCommand" nillable="true" type="soapenc:string"/>
    <element name="cliPassword" nillable="true" type="soapenc:string"/>
    <element name="cliUserName" nillable="true" type="soapenc:string"/>
    <element name="clientClasses" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="collectBackupSubnets" type="xsd:boolean"/>
    <element name="collectionPassword" nillable="true" type="soapenc:string"/>
    <element name="collectionPort" nillable="true" type="soapenc:int"/>
    <element name="collectionType" nillable="true" type="soapenc:string"/>
    <element name="collectionUser" nillable="true" type="soapenc:string"/>
    <element name="configPath" nillable="true" type="soapenc:string"/>
    <element name="ddns" nillable="true" type="soapenc:string"/>
    <element name="defaultThreshold" nillable="true" type="soapenc:int"/>
    <element name="endExtension" nillable="true" type="soapenc:string"/>
    <element name="failoverIpAddress" nillable="true" type="soapenc:string"/>
    <element name="failoverPeers" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="failoverPort" nillable="true" type="soapenc:int"/>
    <element name="globalSync" type="xsd:boolean"/>
    <element name="id" nillable="true" type="soapenc:int"/>
    <element name="ipAddress" nillable="true" type="soapenc:string"/>
    <element name="leasePath" nillable="true" type="soapenc:string"/>
```

```
    <element name="name" nillable="true" type="soapenc:string"/>
    <element name="optionSet" nillable="true" type="soapenc:string"/>
    <element name="policySet" nillable="true" type="soapenc:string"/>
    <element name="primaryPeers" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="product" nillable="true" type="soapenc:string"/>
    <element name="startScript" nillable="true" type="soapenc:string"/>
    <element name="stopScript" nillable="true" type="soapenc:string"/>
    <element name="v4V6Both" nillable="true" type="soapenc:string"/>
  </sequence>
</complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
Agent	The name of the IPAM Agent that is responsible for contacting this server.	Yes	-29	Invalid agent: *agent*
beginExtension	The text to be inserted at the start of the configuration file. Separate lines with the sequence "\r\n", which is hexadecimal 0x0D followed by 0x0A.	No		
cliArgs	Arguments for the command executed to collect statistics. In particular, if the server is a CNR DHCP agent, this value should contain the cluster name.	No		
cliCommand	Command to collect DHCP statistics.	No		
cliPassword	Password used by the cliCommand to access the DHCP Server	No		
cliUserName	User Name used by the cliCommand to access the DHCP Server	No		
clientClasses	This is an array of DHCP client class names to be used by the DHCP Server.	No	-116	Invalid Client Class: *class*
collectBackup Subnets	Set to TRUE if the collection should also process backup subnets from the server. Defaults to false.	No		
collectionPassword	The password used by the collection method (scp or ftp) to log in to the remote server. Used in conjunction with the 'User name for collection'.	Yes		
collectionPort	The port number the collection method (scp or ftp) is listening on. If no value is specified, this will default to **22** if the collection method is scp, and **21** if the collection method is ftp.	No	-32	Invalid collection port: *port*
collectionType	The method by which the IPAM Agent will collect data from the Network Service. Accepted values are **scp** or **ftp**.	Yes	-31	Invalid collection type: *type*
collectionUser	The username used by the collection method (scp or ftp) to log in to the remote server.	Yes		
configPath	The path to the configuration file of the DHCP server.	Yes		
ddns	Specify 'interim', 'standard', or 'lastin' to enable dynamic DNS updates when this server issues a lease. Defaults to 'none'.	No		

Element	Accepted Values	Required	Return Code	Faultstring
defaultThreshold	Default scope utilization warning threshold. Provide warnings when usage of a pool assigned to this service is exceeded. If no value is specified, this will default to **90**.	No	-57	Invalid alert threshold: *value*
endExtension	The text to be appended to the configuration file. Separate lines with "\r\n", which is hexadecimal 0x0D and 0x0A, respectively.			
failoverIpAddress	IP Address used by this DHCP Server for failover communications.	No		
failoverPort	Port used by this DHCP Server for failover communications.	No		
globalSync	Whether or not to include this server in the Global Sync process. Accepted values are **True** or **False** (case insensitive).	No		
id	The internal ID of this server, as provided by the getDhcpServer call. If this is set, the server is updated instead of added.	No		
ipAddress	The IP address or fully-qualified domain name (FQDN) of the Network Service. This must be a valid IPv4 or IPv6 IP address, or a fully-qualified host name.	Yes	-26 -112	Invalid IP address: *address* Duplicate IP Address
name	The name of the Network Service. This can be any combination of letters and numbers.	Yes	-112	Duplicate name
optionSet	The name of a DHCP option set defined in the system to be used by this server.	No	-67	Option Set *name* not found
policySet	The name of a DHCP policy set defined in the system to be used by this server.	No	-68	Policy Set *name* not found
primaryPeers	Not used			
product	The type of DHCP server being defined.	Yes	-123	Unknown DHCP Product: *name*
startScript	The full path of the script that starts the server.	No		
stopScript	The full path of the script that stops the server.	No		

Element	Accepted Values	Required	Return Code	Faultstring
v4V6Both	'v4', 'v6' or 'both'	Yes	-191	Invalid value *v4V6Both* specified for V4V6Both.Valid values are 'V4', 'V6' or 'Both' Or Product *product* does not support IP version '*v4V6Both*'

Other returnCodes and faultstrings

Return Code	Faultstring
-1	Unable to obtain session or Rollback failed. Additional information will appear in the web service log.

ImportDomain

Overview

The **ImportDomain** API enables the web service client to import domains into IPAM.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **importDomain** request and response messages.

```
<wsdl:message name="importDomainResponse" />
<wsdl:message name="importDomainRequest">
  <wsdl:part name="inpDomain" type="tns1:WSDomain" />
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSDomain**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSDomain

Below is the portion of *Imports.wsdl* that describes **WSDomain**, the parameter structure passed to **importDomain**. The elements are described in the table that follows.

```
<complexType name="WSDomain">
  <sequence>
    <element name="contact" nillable="true" type="soapenc:string"/>
    <element name="defaultTTL" nillable="true" type="soapenc:string"/>
    <element name="delegated" type="xsd:boolean"/>
    <element name="derivative" nillable="true" type="soapenc:string"/>
    <element name="domainName" nillable="true" type="soapenc:string"/>
    <element name="domainType" nillable="true" type="soapenc:string"/>
    <element name="expire" nillable="true" type="soapenc:string"/>
    <element name="id" nillable="true" type="soapenc:int"/>
    <element name="infoTemplate" nillable="true" type="soapenc:string"/>
    <element name="managed" type="xsd:boolean"/>
    <element name="negativeCacheTTL" nillable="true" type="soapenc:string"/>
    <element name="refresh" nillable="true" type="soapenc:string"/>
    <element name="retry" nillable="true" type="soapenc:string"/>
    <element name="reverse" type="xsd:boolean"/>
    <element name="serialNumber" nillable="true" type="soapenc:long"/>
    <element name="templateDomain" nillable="true" type="soapenc:string"/>
    <element name="userDefinedFields" nillable="true" type="impl:ArrayOf_soapenc_string"/>
  </sequence>
</complexType>
```

Element	Description and accepted values	Required	Return Code	Faultstring
contact	The contact email address in dotted format. For example, an email address of 'root@ins.com', would be represented as 'root.ins.com'. If not specified a default contact name will be formed by prepending 'dnsadmin' to the domain name as in 'dnsadmin.ins.com.'.	no		
defaultTTL	Default time to live. If not specified, defaults to "86400"; ignored if Managed is "False".	no		
delegated	Indicates that this domain will be associated directly with a zone file. Accepted values are **true** or **false**. If not specified, defaults to **true**.	no		
derivitive	Specify the role of this domain. This can be one of "STANDARD", "TEMPLATE", or "ALIAS". If not specified, defaults to "STANDARD". The *delegated* flag MUST be set to true in order to import a ""TEMPLATE" or "ALIAS" domain.	no	-92 -334	Invalid derivative type: *derivativeType* *delegated* flag is false
domainName	The name of the domain being created.	yes	-95 -135 -157	Invalid domain name Domain name required Domain already exists
domainType	Domain type name already defined to IPAM. If not specified, the "Default" domain type will be used. When importing an "ALIAS" domain its domain type is used to help locate the "TEMPLATE" domain it will be aliased to. Therefore they MUST match for the template domain to be found.	no	-81	Domain type not found: *domainType*
expire	Zone expire time. If not specified, defaults to "604800"; ignored if Managed is "False".	no		
id	The internal ID of this domain, for future use.	no		
infoTemplate	The name of the information template to be associated with this container.	no	-113	Information template not found: *infoTemplateName*
managed	Indicates that this domain is fully defined in IPAM. Accepted values are **true** or **false**. If not specified, defaults to **true**.	no		
negativeCacheTTL	Zone negative cache time to live. If not specified, defaults to "86400"; ignored if Managed is "False".	no		

Element	Description and accepted values	Required	Return Code	Faultstring
refresh	Zone refresh interval. If not specified, defaults to "10800"; ignored if Managed is "False".	no		
retry	Zone retry interval. If not specified, defaults to "3600"; ignored if Managed is "False".	no		
reverse	Indicates that this domain is a reverse in-addr.arpa domain. Accepted values are **true** or **false**. If not specified, defaults to **false**.	no		
serialNumber	Zone serial number. If not specified, defaults to "1"; ignored if Managed is "False".	no		
templateDomain	Name of template domain. When importing an "ALIAS" domain its domain type is used to help locate the named "TEMPLATE" domain it will be aliased to. Therefore they MUST match for the template domain to be found.	yes, if Derivative is "ALIAS"	-60 -135	Template domain not found: *templateDomainName* Template domain name required
userDefinedFields	The user defined fields associated with this domain, as listed in the domain information template specified in parameter infoTemplate. Specify as a string array containing one or more *name*=*value* pairs, where the *name* is the UDF field name/tag and the *value* is the desired value, for example, **State=PA.**	yes, for UDFs defined as required fields	-61 -63 -64	Information Template UDF not found: *udf* Invalid UDF: *udf* Missing required UDF: *udf*

Other returnCodes and faultstrings

Return Code	Faultstring
-1	various unexpected DB errors
-2	Allocation failed
-3	Invalid arguments missing xxx parameter
-99	access denied

ImportDomainResourceRecord

Overview

The **ImportDomainResourceRecord** API enables the web service client to import resource records into IPAM that are not bound to a device, but still appear in a zone file.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **importDomainResourceRecord** request and response messages.

```
<wsdl:message name=" importDomainResourceRecordResponse" />
<wsdl:message name="importDomainResourceRecordRequest">
  < <wsdl:part name="inpRR" type="tns2:WSDomainResourceRec"/>
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSDomainResourceRec**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSDomainResourceRec

Below is the portion of *Imports.wsdl* that describes **WSDomainResourceRec**, the parameter structure passed to **importDomainResourceRecord**. The elements are described in the table that follows.

```
<complexType name=" WSDomainResourceRec ">
  <sequence>
    <element name="TTL" nillable="true" type="soapenc:string" />
    <element name="comment" nillable="true" type="soapenc:string" />
    <element name="data" nillable="true" type="soapenc:string" />
    <element name="deviceRecFlag" type="xsd:boolean"/>
    <element name="domain" nillable="true" type="soapenc:string" />
    <element name="domainType" nillable="true" type="soapenc:string" />
    <element name="id" nillable="true" type="soapenc:int"/>
    <element name="owner" nillable="true" type="soapenc:string" />
    <element name="resourceRecClass" nillable="true" type="soapenc:string" />
    <element name="resourceRecType" nillable="true" type="soapenc:string" />
  </sequence>
</complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
id	The internal identifier for this resource record. If set and a matching record with this id is found, the object will be updated. If not set, a new resource record object will created.	No	-124	No resource record with input RR id: *id*
domainType	Domain type already defined to IPAM. If not specified, the "Default" domain type will be used.	No	-108	Domain not found: domainType/domain
domain	Domain name where resource records are to be added.	Yes.	-108 -107 N/A	Domain not found: domainType/domain Domain not specified Not authorized
owner	The owner field of the resource record.	Yes	-89 -106	Owner not specified Invalid character in owner
resourceRecClass	The Class of the resource record. Defaults to "IN".	No		
resourceRecord Type	The Type of the resource record.	Yes	-92	Invalid Type
TTL	The Time To Live for the record.	No		
data	The data portion of the resource record. The format is dependent on the type specified above.	Yes	-91	Data Required
comment	Comment text associated with the resource record.	No		
deviceRecFlag	When **true**, this indicates that the resource record is bound to a device. When **false**, this indicates that the resource record is associated with the domain only, and not a specific device. Valid on export only.	No		

ImportGalaxyDomain

Overview

The **ImportGalaxyDomain** API enables the web service client to assign domains to galaxies in IPAM.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **importGalaxyDomain** request and response messages.

```
<wsdl:message name=" importGalaxyDomainResponse" />
<wsdl:message name="importGalaxyDomaiRequest">
  < <wsdl:part name="inpGalaxyDomain" type="tns2:WSGalaxyDomain"/>
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSGalaxyDomain**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSGalaxyDomain

Below is the portion of *Imports.wsdl* that describes **WSGalaxyDomain**, the parameter structure passed to **importGalaxyDomain**. The elements are described in the table that follows.

```
<complexType name="WSGalaxyDomain">
  <sequence>
    <element name="domainName" nillable="true" type="soapenc:string" />
    <element name="domainType" nillable="true" type="soapenc:string" />
    <element name="galaxyName" nillable="true" type="soapenc:string" />
    <element name="view" nillable="true" type="soapenc:string" />
  </sequence>
</complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
domainName	Domain name, already defined in IPAM, to assign to the specified galaxy.	Yes.	-60 -71 -155 -157 -157	Domain not found: *domain* for domainType: *domainType* Domain Name is required Exception saving galaxy domain *galaxyName* Found problem with profile master *server* : Cannot have the same zone on both the server and the galaxy. Either remove the server from the GalaxyProfile or remove the Zone from the Server. Domain view combination already exists in this galaxy
domainType	The name of the domain type to which the domain belongs. If not specified, the "Default" domain type will be used.	No	-81	Domain type not found: *domainType*
galaxyName	Name of the galaxy to which to assign this domain.	Yes	-153 -154 -156	Galaxy not found: *galaxy* Galaxy must have profile defined Galaxy name required
view	The name of the galaxy view to which to assign this domain.	No	-58	View not found: *view* for galaxy: *galaxy*

ImportNetElement

Overview

The **`ImportNetElement`** API enables the web service client to import network elements into IPAM.

Note: This CLI has been deprecated as of IPAM 8.1.2. Use the **ImportNetworkElement** API instead. ImportNetElement support will be removed in a future release.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **`importNetElement`** request and response messages.

```
<wsdl:message name="importNetElementRequest">
  <wsdl:part name="elementArray" type="impl:ArrayOf_soapenc_string" />
</wsdl:message>
<wsdl:message name="importNetElementResponse">
  <wsdl:part name="importNetElementReturn" type="soapenc:string" />
</wsdl:message>
```

Response

The string returned in the response contains the name of the network element allocated, for example, "Saved netelement: *name*".

Request

The string array that is passed as input from the client to the web service is described in the next section.

Parameters

String Array

The elements of the string array are described below. Note that the **`returnCode`** tag is not used for this call. When an error occurs, the faultcode will be **`Server.userException`**, with the faultstrings shown below.

Element	Description	Accepted Values	Required	Return Code	Faultstring
0	Name	The name of the Network Element. This can be any combination of letters and numbers.	Yes		Duplicate name for Netelement: *name*
1	IP Address	The IP address or fully-qualified domain name (FQDN) of the Network Element. This must be a valid IPv4 or IPv6 IP address, or a full-qualified host name.	Yes		
2	Vendor	The vendor of the Network Element. Vendor must be predefined in IPAM. If not	Yes when Model is specified.		no models found for given description:

Element	Description	Accepted Values	Required	Return Code	Faultstring
		specified, defaults to **Unknown**.			*description* deviceName: *name*, vendorName: *vendor*
3	Model	The model name of the Network Element. Model must be predefined in IPAM. If not specified, defaults to **Unknown**.	Yes when Vendor is specified		same as above
4	Type	The type of Network Element. Accepted values are **cmts**, **router**, switch or **vpn**.	Yes		same as above
5	Global Sync	Whether or not to include this Network Element in the Global Sync process. Accepted values are **True** or **False** (case insensitive).	Yes		
6	Agent Name	The exact name of the IPAM Agent that is responsible for contacting this Network Service.	Yes		
7	Telnet User	A user name used to telnet to this device.	No		
8	Telnet Password	A password used by the telnet user to telnet to this device.	No		
9	Enable Password	Password used to enter "enabled" or "privileged" mode on the device.	No		
10	Read Community String	The community string used by SNMP to read details from this network element.	No		
11	Interface List	Separated by vertical bars ("|").	No		
12	V3 Username	Required if using SNMP V3.	No		SNMP V3 Username required
13	V3 Authentication Protocol	Either **MD5** or **SHA1**. Leave blank or set to **NONE** if no authentication.	No		Unknown authentication protocol: *protocol*
14	V3 Authentication Password	Required if field N is set to either **MD5** or **SHA1**	No		Authentication Password required
15	V3 Privacy Protocol	Only **DES** supported at this time. Leave blank or set to **NONE** if no privacy.	No		Unknown privacy protocol: *protocol* Privacy not allowed without authentication
16	V3 Privacy Password	Required if field P is set to **DES**.	No		Privacy Password required
17	V3 Context	SNMP V3 Context name, if	No		

Element	Description	Accepted Values	Required	Return Code	Faultstring
	Name	needed.			
18	V3 Engine ID	SNMP V3 Engine ID, if needed.	No		

ImportNetworkElement

Overview

The **`ImportNetworkElement`** API enables the web service client to import network elements into IPAM. It deprecates the ImportNetElement API by by adding support for new fields and capabilities, most notably multiple agents and the ability to apply device interface templates on import. The API also supports updating network elements that already exist in the system

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **`importNetworkElement`** request and response messages.

```
<wsdl:message name="importNetworkElementResponse" />
<wsdl:message name="importNetworkElementRequest">
  <wsdl:part name="inpNetworkElement" type="tns2:WSNetworkElement"/>
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **`WSNetworkElement`**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSNetworkElement

Below is the portion of *Imports.wsdl* that describes **`WSNetworkElement`**, the parameter structure passed to **`importNetworkElement`**. The elements are described in the table that follows.

```
<complexType name="WSNetworkElement">
    <sequence>
     <element name="SNMPRetries" nillable="true" type="soapenc:int"/>
     <element name="SNMPTimeout" nillable="true" type="soapenc:int"/>
     <element name="SNMPVersion" nillable="true" type="soapenc:string"/>
     <element name="SNMPsysDescr" nillable="true" type="soapenc:string"/>
     <element name="SNMPsysLocation" nillable="true" type="soapenc:string"/>
     <element name="SNMPsysName" nillable="true" type="soapenc:string"/>
     <element name="SNMPsysServices" nillable="true" type="soapenc:string"/>
     <element name="agentNames" nillable="true" type="impl:ArrayOf_soapenc_string"/>
     <element name="description" nillable="true" type="soapenc:string"/>
     <element name="globalSync" type="xsd:boolean"/>
     <element name="id" nillable="true" type="soapenc:int"/>
     <element name="interfaceNames" nillable="true" type="impl:ArrayOf_soapenc_string"/>
     <element name="interfaceStatus" nillable="true" type="impl:ArrayOf_soapenc_string"/>
     <element name="interfaceTemplate" nillable="true" type="soapenc:string"/>
     <element name="ipAddress" nillable="true" type="soapenc:string"/>
     <element name="model" nillable="true" type="soapenc:string"/>
     <element name="name" nillable="true" type="soapenc:string"/>
     <element name="readCommunityString" nillable="true" type="soapenc:string"/>
     <element name="type" nillable="true" type="soapenc:string"/>
     <element name="v3AuthPassword" nillable="true" type="soapenc:string"/>
     <element name="v3AuthProtocol" nillable="true" type="soapenc:string"/>
     <element name="v3ContextName" nillable="true" type="soapenc:string"/>
     <element name="v3EngineId" nillable="true" type="soapenc:string"/>
     <element name="v3PrivacyPassword" nillable="true" type="soapenc:string"/>
     <element name="v3PrivacyProtocol" nillable="true" type="soapenc:string"/>
     <element name="v3Username" nillable="true" type="soapenc:string"/>
     <element name="vendor" nillable="true" type="soapenc:string"/>
    </sequence>
   </complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
id	The internal identifier for this Network Eement. If set and a matching Network Element with this id is found, the object will be updated. If not set, a new Network Element object will created.	No	-33	Network element not found.
name	The name of the Network Element. This can be any combination of letters and numbers.	Yes	-114 -204 -206	Name required. Invalid name. Duplicate name.
description	A description of the Network Element. Use "\n" to separate lines.	No		
ipAddress	The IP address of the Network Element. This must be a valid IPv4 or IPv6 IP address.	No	-26	Invalid address.
type	The Network Element device type. Typical values include **CMTS**, **Router**, **Switch** and **VPN**. If not specified, defaults to **Unknown**	No	-52	Device type not found.
vendor	The vendor of the Network Element. Vendor must be predefined in IPAM. If not specified, defaults to **Unknown**.	Yes; when **Model** is specified.	-319 -222	Vendor required. Vendor not found.
model	The model name of the Network Element. Model must be predefined in IPAM. If not specified, defaults to **Unknown**.	Yes; when **Vendor** is specified	-320 -317 -321	Model required. Model not found. Invalid Type, Vendor, Model combination.
globalSync	Specifies whether or not to include this Network Element in the Global Sync process. Accepted values are **True** or **False** (case insensitive). If not specified, defaults to **False**	No		
agentNames	The exact names of the IPAM Agents, predefined in IPAM, that are responsible for contacting this Network Service. Can be a list.	No	-29 -333	Agent not found. Admin is not authorized to access agents.
interfaceNames	The exact names of interfaces, predefined in IPAM, to assign to this Network Element. Pair these names with their status specs passed in the Interface Status list. Note this method of assigning	No	-18 -331	Invalid interface. Ambiguous interface; when a Interface Template is also specified.

Element	Accepted Values	Required	Return Code	Faultstring
	Network Element interfaces is mutually exclusive to using an existing Interface Template to do the same. Trying to do it both ways will return an ambiguous interface error.			
interfaceStatus	The status of each named interface called out in the Interface Names list. Accepted values are **enabled**, **disabled** and **deploying**. If not specified or unrecognized, defaults to **enabled**.	No		
Interface Template	The name of an Interface Template, predefined in IPAM, to apply when assigning interfaces to this Network Element. Note this method of assigning Network Element interfaces is mutually exclusive to using the Interface Name/Status list fields to do the same. Trying to do it both ways will return an ambiguous interface error.	No	-18 -113 -331	Invalid interface. Interface template not found. Ambiguous interface; when a Interface Name / Status list is also specified.
SNMPVersion	**V1, V2** or **V3**, if not specified, defaults to V1. Note the following V1,V2 and V3 parameters sets are mutually exclusive Specifying both will return an ambiguous SNMP version error.	No	-322 -323	Invalid SNMP version. Ambiguous SNMP version.
readCommunity String	The community string used by SNMP V1, SNMP V2 to read details from this network element.	Yes; when SNMP V2 is specified.	-354	SNMP V2 requires a read community string.
SNMPRetries	Specifies the max number of connection retries the SNMP Agent will attempt.	No		
SNMPTimout	Specifies the number of ms the SNMP Agent will wait without receiving messages from its partner before it assumes that the connection to its partner has failed.	No		
v3Username	SNMP V3 username.	Yes; when SNMP V3 is specified	-325	V3 username required.
v3Auth Protocol	Accepted values include **MD5** or **SHA1**. Leave blank to disable authentication.	No	-326	Unknown authentication protocol.
v3Auth	Required if authentication enabled. (Required on import	Yes; when an auth protocol	-327	Authentication

Element	Accepted Values	Required	Return Code	Faultstring
Password	only; optional on overwrite.)	is specified.		password required
v3Privacy Protocol	Accepted values include **AES** or **DES.** Leave blank to disable privacy protection. Also requires authentication to be active to be recognized.	No	-328	Unknown privacy protocol.
v3Privacy Password	Required if privacy protection enabled. (Required on import only; optional on overwrite.)	Yes; when an auth and priv protocol is specified.	-329	Privacy password required
v3Context Name	Optional SNMP context name.	No		
v3EngineId	Optional SNMP engine ID.	No		
SNMPSysName	This is a read only, MIB-II, parameter populated after a IPAM Discover Router Subnets task is run against a Network Element. See RFC-1213 for more details.	Ignored		
SNMPSysDescr	This is a read only, MIB-II, parameter populated after a IPAM Discover Router Subnets task is run against a Network Element. See RFC-1213 for more details	Ignored		
SNMPSys Location	This is a read only, MIB-II, parameter populated after a IPAM Discover Router Subnets task is run against a Network Element. See RFC-1213 for more details	Ignored		
SNMPSys Services	This is a read only, MIB-II, parameter populated after a IPAM Discover Router Subnets task is run against a Network Element. See RFC-1213 for more details	Ignored		

Other returnCodes and faultstrings

Condition	Return Code
Method passed null input argument.	-128
Admin is not authorized to access Network Element functions.	-318

ImportNetElementInterface

Overview

The **ImportNetElementInterface** API enables the web service client to import Network Element Interfaces into IPAM.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **importNetElementInterface** request and response messages.

```
<wsdl:message name="importNetElementInterfaceResponse" />
<wsdl:message name="importNetElementInterfaceRequest">
  <wsdl:part name="inpNetElementInterface" type="tns2:WSNetElementInterface" />
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSNetElementInterface**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSNetElementInterface

Below is the portion of *Imports.wsdl* that describes **WSNetElementInterface**, the parameter structure passed to **importNetElementInterface**. The elements are described in the table that follows.

```
<complexType name="WSNetElementInterface">
  <sequence>
    <element name="id" nillable="true" type="soapenc:int"/>
    <element name="interfaceName" nillable="true" type="soapenc:string"/>
    <element name="netElementName" nillable="true" type="soapenc:string"/>
    <element name="status" nillable="true" type="soapenc:string"/>
  </sequence>
</complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
id	The internal identifier for this network element interface object. If this is not set, a new interface is created. If this is set, the interface with the matching identifier is updated.	No		

Element	Accepted Values	Required	Return Code	Faultstring
interfaceName	The name of the interface being added or modified.	Yes	-37 -20	Interface already exists with this name: *name* for netelement: *netelement* Interface name is required
netElementName	The name of a Network Element already defined to IPAM.	Yes	-33 -35	Network element not found: *netelement* Element name required
status	The status of the interface. This can be one of "Disabled", "Enabled", or "Deployed". The default on an import is "Enabled".	No	-34	Invalid interface status: *status*

Other returnCodes and faultstrings

Return Code	Faultstring
-2	Exception occurred trying to read the network element.

ImportNetService

Overview

The **ImportNetService** API enables the web service client to import DHCP network services into IPAM.

Note: This API has been deprecated as of IPAM 8.1.2. Use the **ImportDhcpServer** API instead. ImportNetService support will be removed in a future release.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **importNetService** request and response messages.

```
<wsdl:message name="importNetServiceResponse" />
<wsdl:message name="importNetServiceRequest">
  <wsdl:part name="inpNetService" type="tns1:WSNetService" />
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSNetService**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSNetService

Below is the portion of *Imports.wsdl* that describes **WSNetService**, the parameter structure passed to **importNetService**. The elements are described in the table that follows.

```
<complexType name="WSNetService">
  <sequence>
    <element name="agentName" nillable="true" type="soapenc:string" />
    <element name="collectionMethod" nillable="true" type="soapenc:string" />
    <element name="collectionPort" nillable="true" type="soapenc:string" />
    <element name="container" nillable="true" type="impl:ArrayOf soapenc_string" />
    <element name="globalSync" nillable="true" type="soapenc:string" />
    <element name="ipAddress" nillable="true" type="soapenc:string" />
    <element name="name" nillable="true" type="soapenc:string" />
    <element name="threshold" nillable="true" type="soapenc:string" />
    <element name="type" nillable="true" type="soapenc:string" />
    <element name="userName" nillable="true" type="soapenc:string" />
    <element name="userPassword" nillable="true" type="soapenc:string" />
    <element name="vendor" nillable="true" type="soapenc:string" />
    <element name="vendorInfo" nillable="true" type="impl:ArrayOf_soapenc_string"/>
  </sequence>
</complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
agentName	The name of the IPAM Agent that is responsible for contacting this Network Service.	Yes	-29	Invalid network service agent: *agent*
collectionMethod	The method by which the IPAM Agent will collect data from the Network Service. Accepted values are **scp** or **ftp**.	No	-31	Invalid network service collection method: *method*
collectionPort	The port number the collection method (scp or ftp) is listening on. If no value is specified, this will default to **22** if the collection method is scp, and **21** if the collection method is ftp.	No	-32	Invalid network service collection port: *port*
container	No longer used.			
globalSync	Whether or not to include this Network Service in the Global Sync process. Accepted values are **True** or **False** (case insensitive). If no value is specified, this will default to **False**.	No	-30	Invalid network global sync option: *option*
ipAddress	The IP address or fully-qualified domain name (FQDN) of the Network Service. This must be a valid IPv4 or IPv6 IP address, or a fully-qualified host name.	Yes	-26	Invalid network service address: *address*
name	The name of the Network Service. This can be any combination of letters and numbers.	Yes		
threshold	Default scope utilization warning threshold. Provide warnings when usage of a pool assigned to this service is exceeded. If no value is specified, this will default to **90**.	No	-57	Invalid alert threshold: *value*
type	The type of Network Service. Accepted value is **dhcp**. If this column is left blank, **dhcp** is assumed.	No	-27	Invalid network service type: *type*
userName	The username used by the collection method (scp or ftp) to log in to the remote server.	No		
userPassword	The password used by the collection method (scp or ftp) to log in to the remote server. Used in conjunction with the 'User name for collection'.	No		
vendor	The Network Service product name. This must be a value already defined in IPAM, for example, **CNR DHCP**.	Yes	-47 -99 -28	productAction failed Product not found Unrecognized collection type: *type*

Element	Accepted Values	Required	Return Code	Faultstring
vendorInfo	A string array containing vendor specific information for the product's collection type. For collection types **qip**,**adc**, **msft** and **isc**, the information includes the DHCP Configuration file pathname and DHCP Active Lease file pathname. For example, /opt/qip/dhcp/dhcpd.conf /opt/qip/dhcp/dhcp.db or c:\qip\dhcp\dhcpd.conf c:\qip\dhcp\dhcp.db For collection type **cnr**, the information includes the Path/Executable of NRCD command, the NRCMD user id, the NRCMD password and the Cluster Name. For example, /opt/cnr/bin/nrcmd|myuserid|mypass cluster1	No		

Other returnCodes and faultstrings

Return Code	Faultstring
-1	Unable to obtain session or Rollback failed. Additional information will appear in the web service log.

ImportNetworkLink

Overview

The **ImportNetworkLink** API enables the web service client to import and update network links to IPAM. Only logical Network Links can be imported.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **importNetworkLink** request and response messages.

```
<wsdl:message name="importNetworkLinkRequest">
      <wsdl:part name="networkLink" type="tns2:WSNetworkLink"/>
</wsdl:message>

<wsdl:message name="importNetworkLinkResponse">
      <wsdl:part name="importNetworkLinkReturn" type="soapenc:string"/>   </wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSNetworkLink**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSNetworkLink

Below is the portion of *Imports.wsdl* that describes **`WSNetworkLink`**, the parameter structure passed to **`importNetworkLink`**. The elements are described in the table that follows.

```
<complexType name="WSNetworkLink">
    <sequence>
     <element name="blockNames" nillable="true" type="impl:ArrayOf_soapenc_string"/>
     <element name="containers" nillable="true" type="impl:ArrayOf_soapenc_string"/>
     <element name="description" nillable="true" type="soapenc:string"/>
     <element name="dhcpOptionSet" nillable="true" type="soapenc:string"/>
     <element name="dhcpPolicySet" nillable="true" type="soapenc:string"/>
     <element name="id" nillable="true" type="soapenc:string"/>
     <element name="name" nillable="true" type="soapenc:string"/>
     <element name="type" nillable="true" type="soapenc:string"/>
    </sequence>
   </complexType>
```

Element	Description and accepted values	Required	Return Code	Faultstring
blockNames	Ignored on import.	no		
containers	Ignored on import.	no		
description	A description of the network link. Use "\n" to separate lines.	no		
dhcpOptionSet	The name of a DHCP Option Set defined within IPAM that will be applied to subnets using this network link.	no	-67	DHCP Option Set *optionSetName* not found
dhcpPolicySet	The name of a DHCP Policy Set defined within IPAM that will be applied to subnets using this network link.	no	-68	DHCP Policy Set *policySetName* not found
id	The internal ID of this network link, as provided by a GetNetworkLink call. If this is set, the network link is updated instead of added.	yes, for modify	-279	Network Link not found, ID=*id*
Name	The name of the network link.	yes	-287 -288	Network Link name required Duplicate network link name: *name*

Element	Description and accepted values	Required	Return Code	Faultstring
type	The only value accepted for import is the default, **logical**. This element is provided for informational purposes for the ExportNetworkLink and GetNetworkLink APIs.	no	-282 -283	Network link type must be logical or physical Only logical network links may be imported

Other returnCodes and faultstrings

Return Code	Faultstring
-2	Allocation failed
-291	Admin is not authorized to access Network Link functions

ImportPrefixPool

Overview

The **ImportPrefixPool** API enables the web service client to import or modify prefix pools in IPAM.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **importPrefixPool** request and response messages.

```
<wsdl:message name="importPrefixPoolRequest">
  <wsdl:part name="prefixPool" type="tns2:WSPrefixPool"/>
</wsdl:message>
<wsdl:message name="importPrefixPoolResponse"/>
```

Response

There is no data in the response.

Request

The complex type **WSPrefixPool**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSPrefixPool

Below is the portion of *Imports.wsdl* that describes **WSPrefixPool**, the parameter structure passed to **importPrefixPool**. The elements are described in the table that follows.

```
  <complexType name="WSPrefixPool">
 <sequence>
  <element name="allowClientClasses" nillable="true" type="impl:ArrayOf_soapenc_string"/>
  <element name="container" nillable="true" type="soapenc:string"/>
  <element name="delegatedPrefixLength" nillable="true" type="soapenc:int"/>
  <element name="denyClientClasses" nillable="true" type="impl:ArrayOf_soapenc_string"/>
  <element name="dhcpOptionSet" nillable="true" type="soapenc:string"/>
  <element name="dhcpPolicySet" nillable="true" type="soapenc:string"/>
  <element name="id" nillable="true" type="soapenc:int"/>
  <element name="length" nillable="true" type="soapenc:int"/>
  <element name="longestPrefixLength" nillable="true" type="soapenc:int"/>
  <element name="name" nillable="true" type="soapenc:string"/>
  <element name="overlapInterfaceIp" type="xsd:boolean"/>
  <element name="primaryNetService" nillable="true" type="soapenc:string"/>
  <element name="shortestPrefixLength" nillable="true" type="soapenc:int"/>
  <element name="startAddr" nillable="true" type="soapenc:string"/>
  <element name="type" nillable="true" type="soapenc:string"/>
 </sequence>
</complexType>
```

ImportPrefixPool

Element	Description and accepted values	Required	Return Code	Faultstring
id	The internal identifier for this address pool object. If this is not set, a new prefix pool is created. If this is set, the prefix pool with the matching identifier is updated.	No for creates, Yes for updates.	-117	PrefixPool ID=<id> not found Invalid ID <id> on prefixPool object
startAddr	The IP Address of the first address in the pool. This address must be in a block with an In-Use/Deployed status.	Yes	-115 -36 -97 -115	Missing Start or Length Block Not Found Block Not Unique Invalid Start Address
Length	Length of the prefix pool	Yes	-258 -26	Missing Prefix Length Bad Prefix Length End Address is Outside of Block
type	One of "Dynamic PD DHCPv6"," Automatic PD DHCPv6"	Yes	-73	Invalid Address Type
name	Address Pool name. Defaults to "Start Address/Length"	No		
container	The name of the container that holds the block in which the pool is defined. This is required only if there is overlapping address space in use and the start address is in overlapping space. The container is then used to uniquely determine the block that will contain the address pool.	No, unless startAddr is not unique.	-36	Block not found in container
delegatedPrefixLength	Specifies the default length of the delegated prefix	Yes	-258	Missing Delegated Prefix Length Bad Delegated Prefix Length
shortestPrefixLength	Specifies the shortest length of the delegated prefix. CNR only.	No		
longestPrefixLength	Specifies the longest length of the delegated prefix. CNR only.	No		
primaryNetService	The name of the DHCP server that will serve prefixes from this pool	No	-94	DHCP Server not found DHCP Server not valid for this container
dhcpOptionSet	The name of an Option Set used with this pool	No	-67	DHCP Option Set not found
dhcpPolicySet	The name of a Policy Set used with this pool.	No	-68	DHCP Policy Set not found

Element	Description and accepted values	Required	Return Code	Faultstring
allowClient Classes	An array of Client Classes that are allowed in this address pools. Each element of the array names a different Client Class.	No	-116	DHCP Client Class not found
denyClientClas ses	An array of Client Classes that are NOT allowed in this address pools. Each array element names a different Client Class.	No	-116	DHCP Client Class not found
overlapInterfa ceIp	Flag to allow a DHCPv6 pool to overlap an interface address. This flag may be set only if pool is managed by a CNR DHCPv6 server. This is ignored for DHCPv4 pools.	No	-344	Invalid DHCP server.

ImportRootBlock

Overview

The **ImportRootBlock** API enables the web service client to import root blocks into IPAM.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **importRootBlock** request and response messages.

```
<wsdl:message name="importRootBlockResponse" />
<wsdl:message name="importRootBlockRequest">
  <wsdl:part name="inpRootBlock" type="tns1:WSRootBlock" />
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSRootBlock**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSRootBlock

Below is the portion of *Imports.wsdl* that describes **WSRootBlock**, the parameter structure passed to **importRootBlock**. The elements are described in the table that follows.

```
<complexType name="WSRootBlock">
  <sequence>
    <element name="RIR" nillable="true" type="soapenc:string" />
    <element name="SWIPname" nillable="true" type="soapenc:string" />
    <element name="allocationReason" nillable="true" type="soapenc:string" />
    <element name="allocationReasonDescription" nillable="true" type="soapenc:string" />
    <element name="allowOverlappingSpace" nillable="true" type="soapenc:string" />
    <element name="blockAddr" nillable="true" type="soapenc:string" />
    <element name="blockName" nillable="true" type="soapenc:string" />
    <element name="blockSize" nillable="true" type="soapenc:string" />
    <element name="blockType" nillable="true" type="soapenc:string" />
    <element name="container" nillable="true" type="soapenc:string" />
    <element name="createReverseDomains" nillable="true" type="soapenc:string" />
    <element name="description" nillable="true" type="soapenc:string" />
    <element name="domainType" nillable="true" type="soapenc:string" />
    <element name="organizationId" nillable="true" type="soapenc:string" />
    <element name="userDefinedFields" nillable="true" type="impl:ArrayOf_soapenc_string" />
  </sequence>
</complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
RIR	The Regional Internet Registry this space was obtained from. Accepted values are: **Generic**, **RFC1918**, **ARIN**, **RIPE**, **APNIC**, **LACNIC**, and **AFRINIC**. If not specified, **Generic** is assumed.	No	-14	Unknown root block type (RIR): *type*
SWIPname	SWIP/Net name for the block.	Yes, if required by Container rules	-66 -70	SWIPname is required for this container/ blocktype SWIPname is not allowed for this container/ blocktype
allocationReason	The name of a pre-existing Allocation Reason.	No	-22	Invalid allocation reason: *reason*
allocationReason Description	A description of the reason for the allocation.	No		
allowOverlapping Space	Whether or not to allow duplicate (overlapping) address space in this block. Accepted values are **true** or **false**. If not specified, defaults to **false**.	No	-159	Overlapping Public Address Space is not allowed.
blockAddr	The IP block to create. This should be in the format of a network address (e.g., 10.0.0.0).	Yes	-12	Invalid block address: *exception*
blockName	A name for the block. Defaults to system supplied name of *Address space/Block size*.	No		
blockSize	The size of the block in short-notation (e.g., 24 for a 255.255.255.0 network).	Yes	-16 -15	Invalid block size mask: *exception* Invalid block size: *exception*
blockType	The Block Type of the block. If not specified, a block type of **Any** is assumed.	No	-13	Unknown block type
container	The name of the logical container that will hold the block. Names can be in either short or long format. Short format example: **Dallas**. Long format example: **InControl/Texas/Dallas**. Long format eliminates ambiguity in cases where there are duplicate container names.	Yes	-2 -6 -7 -5 -1 -5 -148 -277	Could not find container: *container* Invalid container name Ambiguous container name Container not found Database error Unable to lookup container Container does not allow root block creation. Cannot add root block to device container

Element	Accepted Values	Required	Return Code	Faultstring
createReverse Domains	Whether or not to automatically create reverse DNS domain(s) for this block. Accepted values are **true** or **false**. If not specified, defaults to **false**.	No	-82	createReverseDomains must be true or false: *value*
description	A description of the block. Use "\n" to separate lines.	No	-198	Block Description cannot exceed 255 characters
domainType	Specify the domain type for the reverse DNS domain(s) to be created when Create Reverse Domains is "true". If not specified, defaults to "Default".	No	-81	Domain type not found: *domainType*
organizationId	The organization id for the Regional Internet Registry this space was obtained from. This id must be predefined in IPAM.	No	-84	Invalid Organization Id: *id*
userDefinedFields	A string array containing one or more *name=value* pairs, where the *name* is the UDF field name/tag and the *value* is the desired value, for example**, State=PA**. If the UDF type is Checkbox, the valid values are **on** and **off**.	Yes, for UDFs defined as required fields.	-53	store UDFs failed: *SQL exception message*

Other returnCodes and faultstrings

Return Code	Faultstring
-24	Duplicate or overlapping root block: *blockname*
-25	Duplicate or overlapping root block in container tree: *blockname*
-99	Access Denied. Either the administrator does not have rights to add blocks in general OR the administrator does not have rights to add blocks to the specified container.

ImportZone

Overview

The **ImportDnsZone** API enables the web service client to import zones into IPAM.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **importDnsZone** request and response messages.

```
<wsdl:message name="importDnsZoneResponse" />
<wsdl:message name="importDnsZoneRequest">
  <wsdl:part name="inpZone" type="tns1: WSDnsZone" />
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSDnsZone**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSDnsZone

Below is the portion of *Imports.wsdl* that describes **WSDnsZone**, the parameter structure passed to **importDnsZone**. The elements are described in the table that follows.

```
<complexType name="WSDnsZone">
  <sequence>
     <element name="MNAME" nillable="true" type="soapenc:string"/>
     <element name="acceptZoneTransfers" nillable="true" type="soapenc:string"/>
     <element name="aliasZone" type="xsd:boolean"/>
     <element name="allowUpdateACL" nillable="true" type="soapenc:string"/>
     <element name="autogenNSGlue" nillable="true" type="soapenc:string"/>
     <element name="domainName" nillable="true" type="soapenc:string"/>
     <element name="domainType" nillable="true" type="soapenc:string"/>
     <element name="dynamicZone" type="xsd:boolean"/>
     <element name="filename" nillable="true" type="soapenc:string"/>
     <element name="galaxyName" nillable="true" type="soapenc:string"/>
     <element name="masters" nillable="true" type="soapenc:string"/>
     <element name="publishNS" nillable="true" type="soapenc:string"/>
     <element name="server" nillable="true" type="soapenc:string"/>
     <element name="templateZone" type="xsd:boolean"/>
     <element name="updatePolicy" nillable="true" type="soapenc:string"/>
     <element name="updateZone" nillable="true" type="soapenc:string"/>
     <element name="view" nillable="true" type="soapenc:string"/>
     <element name="zoneExtensionsAfter" nillable="true" type="soapenc:string"/>
     <element name="zoneExtensionsPrior" nillable="true" type="soapenc:string"/>
     <element name="zoneType" nillable="true" type="soapenc:string"/>
  </sequence>
</complexType>
```

Element	Description and accepted values	Required	Return Code	Faultstring
MNAME	Specifies an alternative name for the primary or "master" DNS server that DDNS requests should be sent to for this zone. If not specified, no MNAME will be used. This field only applies to master zones. It will be ignored for all other zone types and any zone designated as an Alias zone.	No	-236	MNAME valid only for master zones
acceptZoneTransfers	Accepted values are **true** or **false**. If not specified, defaults to **true**. This field only applies to master zones. It will be ignored for all other zone types.	No		
aliasZone	Indicates if the imported zone is intended to be a alias zone. The linked template zone will be chosen by using the zone assigned to the template domain to which the alias domain is linked. Accepted values are **true** or **false**. If not specified, defaults to **false**.	No	-238	Domain is not an alias: *domain*
allowUpdateACL	Specifies the DNS 'allow-update' address match list a BIND based server uses to filter DDNS update requests against. This free text field only applies to master zone types and is accepted without input validation. Setting this field value to "!BLANK!" on a zone update will serve to completely remove the 'allow-update' option from the zone. This field is ignored for non-master zones. This field is not supported for Microsoft DNS servers.	Yes, when Dynamic Zone is true, one of Allow Update ACL or Update Policy must be specified.	-252	Update option invalid for non-dynamic zone
autogenNSGlue	Accepted values are true or false. If not specified, defaults to true.	No	-241	autogenNSGlue must be true or false, specified: *value*
domainName	Domain name already defined to IPAM	Yes	-60 -71	Domain not found: *domainType/domain* Domain Name is required
domainType	Domain type name already defined to IPAM. If not specified, the "Default" domain type will be used.	No	-81	Domain type not found: *domainType*

Element	Description and accepted values	Required	Return Code	Faultstring
dynamicZone	Accepted values are **true** or **false**. If not specified, defaults to **false**. **true** indicates that this is a dynamic zone. Applicable only for master zones and BIND based servers. When **true**, you must specify either Update Policy or Allow Update ACL. This defaults to **true** for Microsoft DNS servers.	No	-250 -251 -253 -255	Dynamic zones must be master zones Update option required for dynamic zone Only zones associated with BIND servers can be dynamic Cannot specify both allow update and update policy
filename	The name of the zone file that will be generated by IPAM. If not specified, will default to a system-generated value based on zone type.	No		
galaxyName	Future use.	No		
Masters	For zone types **slave** and **stub** only, specify the master server. This field is not valid for other zone types.	Yes for zone types **slave** and **stub**	-237	Masters only valid for slave and stub zones
publishNS	For master and slave zone types, specifies whether the NS record for this zone will be published in the zone file. Defaults to **true** if not specified. This corresponds to the "Include this server in NS record list" checkbox on the zone profile.	No	-241	publishNS must be true or false, specified: *value*
Server	The network service name of the DNS Server.	Yes, if GalaxyName is not supplied.	-235	Server name required
templateZone	Indicates if the imported zone is intended to be a template zone. Accepted values are **true** or **false**. If not specified, defaults to **false**.	No	-239 -240	Template Zone not found: *templateDomain* for server: *server* Domain is not a template: *domain*
updatePolicy	Specifies an update policy, already defined in IPAM, for a dynamic master zone. Setting this field value to "!BLANK!" on a zone update will serve to completely remove the update policy option from the zone. This field is ignored for non-master and non-dynamic zones. This field is not supported for Microsoft DNS servers.	Yes, when Dynamic Zone is true, one of Allow Update ACL or Update Policy must be specified.	-252 -254	Update option invalid for non-dynamic zone Update policy not found: *policy*

Element	Description and accepted values	Required	Return Code	Faultstring
updateZone	Specify **true** to update an existing zone, **false** to import a new zone. If not specified, defaults to **false**. When **true**, specify all fields as you would for an import. Any field not specified on an update is cleared or set to its default value.	No		
View	The name of the view in which the new zone will be created. If specified, the view must exist. If not specified, the new zone will be created in the view named 'Default.'	No	-58	View not found: *view* for server: *server*
zoneExtensionsAfter	Specifies the name of the file containing text lines that will be inserted following the resource records for this zone.	No		
zoneExtensionsPrior	Specifies the name of the file containing text lines that will be inserted prior to the resource records for this zone.	No		
zoneType	Specifies the type of zone being imported. Accepted values are **master**, **slave**, **forward** or **stub**.	Yes	-85 -86	Zone type is required Invalid zone type: *zonetype*

Other returnCodes and faultstrings

Return Code	Faultstring
-2	Various - Allocation failed due to unexpected exception
-99	Access denied
-157	Zone *domainName* already exists. Specify 'Update Zone' to change

JoinBlock

Overview

The **JoinBlock** API enables the web service client to join existing, adjacent blocks into a larger block.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **joinBlock** request and response messages.

```
<wsdl:message name="joinBlockRequest">
    <wsdl:part name="blockName" type="soapenc:string"/>
    <wsdl:part name="container" type="soapenc:string"/>
</wsdl:message>
<wsdl:message name="joinBlockResponse">
</wsdl:message>
```

Response

There is no data in the response.

Request Parameters

blockName

Specify the name of the block, for example, `10.0.0.0/24`. The system searches for non-free blocks by this name.

container

The container name must be specified if the block name is not unique, due to overlapping space or naming conventions.

Return Codes and Fault Strings

Return Code	Fault String
-12	Invalid Block Address
-17	Invalid Block Status
-36	Block not found
-97	Block not unique
-138	Block exists in multiple containers
-139	No valid adjoining block
-140	Block is not on bit boundary
-141	Blocks are not contiguous
-142	Block join size different
-199	Container not found
-53	SQL Exception pinning connection
-99	Access Denied

ModifyBlock

Overview

The **ModifyBlock** API enables the web service client to update certain fields in an existing Block. To modify a block, use this call in conjunction with the **GetBlock** API (Page 3398) calls. First, retrieve the block using a **GetBlock** call. Then, modify the returned structure (see below). Lastly, pass that modified structure to the **ModifyBlock** API.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **modifyBlock** request and response messages.

```
<wsdl:message name="modifyBlockRequest">
    <wsdl:part name="block" type="tns2:WSGenericBlock"/>
    <wsdl:part name="container" type="soapenc:string"/>
</wsdl:message>
<wsdl:message name="modifyBlockResponse">
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSGenericBlock**, which is passed as input from the client to the web service, is described in the next section.

Parameters

Container

The container name must be specified if the block is attached to multiple containers. This enables the User Defined Field updates to be processed correctly.

WSGenericBlock

Below is the portion of *Imports.wsdl* that describes **WSGenericBlock**, the parameter structure passed to **modifyBlock**. The elements are described in the table that follows.

```
<complexType name="WSGenericBlock">
  <sequence>
    <element maxOccurs="unbounded" name="addrDetails" nillable="true"
      type="tns2:WSAllocationTemplateDetails"/>
    <element name="allocationReason" nillable="true" type="soapenc:string"/>
    <element name="allocationReasonDescription" nillable="true" type="soapenc:string"/>
    <element name="allocationTemplateName" nillable="true" type="soapenc:string"/>
    <element name="allowOverlappingSpace" type="xsd:boolean"/>
    <element name="blockAddr" nillable="true" type="soapenc:string"/>
    <element name="blockName" nillable="true" type="soapenc:string"/>
    <element name="blockSize" type="xsd:int"/>
    <element name="blockStatus" nillable="true" type="soapenc:string"/>
    <element name="blockType" nillable="true" type="soapenc:string"/>
    <element name="container" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="createadmin" nillable="true" type="soapenc:string"/>
    <element name="createdate" nillable="true" type="xsd:dateTime"/>
    <element name="description" nillable="true" type="soapenc:string"/>
    <element name="excludeFromDiscovery" nillable="true" type="soapenc:string"/>
    <element name="id" nillable="true" type="soapenc:int"/>
    <element name="inheritDiscoveryAgent" nillable="true" type="soapenc:int"/>
    <element name="interfaceAddress" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="interfaceName" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="ipv6" type="xsd:boolean"/>
    <element name="lastadminid" type="xsd:int"/>
    <element name="lastupdate" nillable="true" type="xsd:dateTime"/>
    <element name="nonBroadcast" type="xsd:boolean"/>
    <element name="numaddressablehosts" type="xsd:long"/>
    <element name="numallocatedhosts" type="xsd:long"/>
    <element name="numassignedhosts" type="xsd:long"/>
    <element name="numdynamichosts" type="xsd:long"/>
    <element name="numleasablehosts" type="xsd:long"/>
    <element name="numlockedhosts" type="xsd:long"/>
    <element name="numreservedhosts" type="xsd:long"/>
    <element name="numstatichosts" type="xsd:long"/>
    <element name="numunallocatedhosts" type="xsd:long"/>
    <element name="organizationId" nillable="true" type="soapenc:string"/>
    <element name="primarySubnet" type="xsd:boolean"/>
    <element name="rir" nillable="true" type="soapenc:string"/>
    <element name="rootBlock" type="xsd:boolean"/>
    <element name="rootBlocktype" nillable="true" type="soapenc:string"/>
    <element name="subnet" nillable="true" type="tns1:WSSubnetPolicy"/>
    <element name="subnetlosshosts" type="xsd:long"/>
    <element name="swipname" nillable="true" type="soapenc:string"/>
    <element name="userDefinedFields" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="ignoreErrors" type="xsd:boolean"/>
  </sequence>
</complexType>
```

Element	Description	Modify?	Return Code	Fault String
addrDetails	Define attributes of any address allocations of the specified allocation template. See below.	Yes		
allocationReason	The architected code for block creation	No		
allocationReason Description	The text entered at creation for the allocation reason.	No.		

ModifyBlock

Element	Description	Modify?	Return Code	Fault String
allocationTemplateName	For a block with blockStatus=**Deployed**, or if changing the blockStatus to **Deployed**, the name of the allocation template to use to create address pools from the newly created block.	Yes	-17	Allocation Template supported only for In-Use/Deployed blocks
			-21	Invalid offset
			-65	Invalid allocation template: *name*
			-177	Address Pool Template detail overlaps interface address.
			-182	No domain set, create resource records failed.
			-189	Default Gateway(s) have been defined in the Subnet Policies. Address pool templates with default gateway option are not allowed.
			-207	Range conflicts with pool *pool* in Block *block name* in Container *container name*
allowOverlappingSpace	If true, the block can overlap other blocks in the system.	No		
blockAddr	The starting address of the block	No		
blockName	The Name of the block	Yes		
blockSize	The CIDR size of the block	No		
blockStatus	Block Status, one of "Free", "Reserved", "Aggregate", "Deployed", "FullyAssigned"	Yes	-17	Invalid Block Status
blockType	The name of the block type.	No		
container	An array of container names	No		
createadmin	The name of the administrator that created the block	No		
createdate	The date and time when this block was created	No		
description	The block Description. Use "\n" to separate lines.	Yes	-198	Block Description cannot exceed 255 characters.

Element	Description	Modify?	Return Code	Fault String
excludeFromDiscovery	Flag indicating if this subnet should be included in Host Discovery tasks. Accepted values are **true** or **false**. If not specified, defaults to **false**. Valid only for Deployed blocks.	Yes	-13 -241	BlockModify set exclude from discovery failed: flag supported only for In-Use/Deployed blocks BlockModify set exclude from discovery must be true or false
id	The block ID. This MUST be supplied in order to update the block.	No	-36	Block not found
inheritDiscoveryAgent	Setting to indicate if the block inherits the discover agent from the container or not	No		
interfaceAddress	Array of IP Addresses where this block connects to a network element. Note that you can modify or delete an **Interface** type address, but you cannot add one. Note that you cannot modify or delete a virtual interface address. Note that if you are changing the status of a block in a device container from **Reserved** to **Deployed**, an interface offset of 1 is assigned.	No	-273 -275	Error saving interface address: *address* Modifying or deleting a virtual address interface is not supported
interfaceName	Array of Interface Names attached to this block	No		
ipv6	True if this is an IPV6 block	No		
lastadminid	ID of the last administrator to update this block. Updated automatically by this call.	No		
lastupdate	Time of last update to this block. Updated automatically by this call.	No		
nonBroadcast	Flag indicating if this is a Non-Broadcast subnet. Non-broadcast subnets are allowed to assign devices to the subnet and broadcast addresses. Accepted values are **true** or **false.** **Valid for IPv4 Deployed blocks only.**	No		
numaddressablehosts	Number of Addressable hosts	No		
numallocatedhosts	Number of Allocated hosts	No		
numassignedhosts	Number of Assigned hosts	No		

ModifyBlock

Element	Description	Modify?	Return Code	Fault String
numdynamichosts	Number of Dynamic hosts	No		
numleasablehosts	Number of hosts that can be leased	No		
numlockedhosts	Number of locked hosts	No		
numreservedhosts	Number of reserved hosts	No		
numstatichosts	Number of static hosts	No		
numunallocatedhosts	Number of Unallocated Hosts	No		
organizationId	Org ID of the RIR Organization	Yes	-84	Unknown RIR Organization
primarySubnet	Flag indicating if this subnet is primary. Accepted values are **true** or **false**. **Valid only for Deployed blocks in device containers.**	Yes	-17	modifyBlock set primary subnet failed: flag supported only for In-Use/Deployed blocks
rir	Name of the RIR Organization	Yes	-84	Unknown RIR Organization
rootBlock	True if this is a root block	No		
rootBlocktype	Name of the internet registry	Yes	-14	Invalid Root Block Type
subnet	Subnet parameters. See below.	Yes	-17	Subnet policies supported only for In-Use/Deployed blocks
subnetlosshosts	Number of hosts lost due to subnetting	No		
swipname	SWIP Name for ARIN blocks	Yes	-137 -66 -70	Duplicate SWIP Name SWIP Name Required SWIP Name not allowed
userDefinedFields	Array of user defined fields. Each element in the array has the format "name=value" where "name" is the UDF tag name.	Yes	-63 -64	Invalid UDF Missing Required UDF
ignoreErrors	Flag, when set to true will cause the system to ignore any errors associated with reparenting a block from a container with a container/blocktype information template to a container without.	No		

Reparenting a Block via Modify Block API

A block may be reparented by specifying a target container name different than the current parent container name. In the case of reparenting blocks contained in device containers, a target interface name must also be provided. To reparent a block, use this call in conjunction with the **GetBlock** API (page 339) calls. First, retrieve the block using a **GetBlock** call. Then, modify the returned structure, setting the new container name and in the case of device container, the new interface name. Lastly, pass that modified structure to the **ModifyBlock** API.

Note: ModifyBlock either reparents, when a new container name different than the current parent container has been specified, **or** modifies the block without a reparent. Both cannot be performed during the same modify block invocation.

Modifying Block Subnet Policies

Use the subnet field and **WSSubnetPolicy** structure to specify changes to block subnet policies.

WSSubnetPolicy

Below is the portion of *Imports.wsdl* that describes **WSSubnetPolicy**, the structure passed in **WSGenericBlock** as "subnet". The elements are described in the table that follows.

```
<complexType name="WSSubnetPolicy">
  <sequence>
    <element name="DHCPOptionsSet" nillable="true" type="soapenc:string" />
    <element name="DHCPPolicySet" nillable="true" type="soapenc:string" />
    <element name="DNSServers" nillable="true" type="impl:ArrayOf_soapenc_string" />
    <element name="cascadePrimaryDhcpServer" nillable="true" type="xsd:boolean" />
    <element name="defaultGateway" nillable="true" type="soapenc:string" />
    <element name="failoverDHCPServer" nillable="true" type="soapenc:string" />
    <element name="forwardDomainTypes" nillable="true" type="impl:ArrayOf soapenc string"/>
    <element name="forwardDomains" nillable="true" type="impl:ArrayOf_soapenc_string" />
    <element name="networkLink" nillable="true" type="soapenc:string"/>
    <element name="primaryDHCPServer" nillable="true" type="soapenc:string" />
    <element name="primaryWINSServer" nillable="true" type="soapenc:string" />
    <element name="reverseDomainTypes" nillable="true" type="impl:ArrayOf soapenc string"/>
    <element name="reverseDomains" nillable="true" type="impl:ArrayOf_soapenc_string" />
  </sequence>
</complexType>
```

Element	Description	Modify ?	Return code	Fault String
DHCPOptionsSet	The name of a previously defined DHCP Options set.	yes	-67	DHCP Option Set *name* not found
DHCPPolicySet	The name of a previously defined DHCP policy set.	yes	-68	DHCP Policy Set *name* not found
DNSServers	The name of previously defined DNS servers to be sent as an IP address to the client.	yes	-93 -99	DNS Server *name* not found. Admin does not have read access to server: *name*

Element	Description	Modify ?	Return code	Fault String
cascadePrimary DhcpServer	If address pool or individual IP address objects within the subnet reference a specific DHCP server, this attribute can be used to allow the 'primaryDHCPServer' attribute value to "cascade" to these address pool and IP address objects. Note that address pool and IP address objects that are configured with "Same as Subnet" for the primary DHCP server will be unaffected.	yes		
defaultGateway	The default gateway that DHCP clients on this subnet will use. Accepted values are an IP address, or a comma-separate list of IP addresses.	yes	-26	Invalid IP Address *address*
failover DHCPServer	The name of the DHCP server that will act as failover for this subnet. This cannot be the same as the primary DHCP server.	yes	-94 -99	DHCP *name* not found Admin does not have write access to server: *name*
forwardDomains	The forward domain names that will available to the user when adding an IP address to the system. The first forward domain in the list will be used when there is a domain name DHCP option.	yes	-95	Invalid forward DNS Domain *name*
forwardDomain Types	The domain types corresponding to the domains listed in forwardDomains. Only required for non-default domain types.	yes	-95	Invalid forward DNS Domain *name*
networkLink	**Valid for deployed blocks in logical containers only**, the name of a logical network link already defined in IPAM. This is the name of the Shared Network Segment for this subnet. This value is automatically assigned for blocks in device containers.	Yes, for blocks in logical containers only	-279 -280	Network Link not found, name=*name* Network link valid only for logical containers
primaryDHCP Server	The name of the DHCP server that will act as primary for this subnet.	yes	-94 -99	DHCP *name* not found Admin does not have write access to server: *name*
primaryWINS Server	The IP address of the Microsoft WINS server for clients in this subnet.	yes	-26	Invalid IP Address *address*
reverseDomains	The reverse domain names that will available to the user when adding an IP address to the system.	yes	-95	Invalid reverse DNS Domain *name*
reverseDomain Types	The domain types corresponding to the domains listed in reverseDomains. Only required for non-default domain types.	yes	-95	Invalid reverse DNS Domain *name*

Applying an Allocation Template

Use the **addrDetails** field to optionally specify attributes of any address allocations within the allocation template specified by **allocationTemplateName**. You can apply a template and specify address details when changing a block's status to Deployed, or when modifying a block that is already of status Deployed.

WSAllocationTemplateDetails

Below is the portion of *Imports.wsdl* that describes **WSAllocationTemplateDetails**, the structure passed in **WSGenericBlock** as **addrDetails**. The elements are described in the table that follows.

```
<complexType name=" WSAllocationTemplateDetails">
  <sequence>
    <element name="netserviceName" nillable="true" type="soapenc:string"/>
    <element name="offsetFromBeginningOfSubnet" type="xsd:boolean"/>
    <element name="~~sharename~~" nillable="true" type="soapenc:string"/>
    <element name="startingOffset" type="xsd:long"/>
  </sequence>
</complexType>
```

Element	Description	Locator/ Modify?	Return Code	Fault String
netserviceName	The name of the network service for this address allocation	No/Yes	-185	Invalid network service name: *name*
offsetFrom BeginningOf Subnet	Specify **true** or **false**. This must match the specification in the Allocation Template. If not specified, defaults to false.	Yes/No		
~~sharename~~ *Deprecated*	~~The name used to link address pools together.~~	No		
startingOffset	Identify the address allocation within the template. This must match the specification in the Allocation Template..	Yes/No	-174	Invalid starting offset: *offset* for allocation template: *templateName*

ModifyPendingApproval

Overview

The **ModifyApproval** API enables the web service client to approve or reject changes submitted to the administrator's pending approval queue.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **modifyPendingApproval** request and response messages.

```
<wsdl:message name="modifyPendingApprovalRequest">
  <wsdl:part name="approval" type="tns2:WSPendingApproval"/>
</wsdl:message>
<wsdl:message name="modifyPendingApprovalResponse">
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSPendingApproval**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSPendingApproval

Below is the portion of *Imports.wsdl* that describes **WSPendingApproval**, the parameter structure passed to **modifyPendingApproval**. The elements are described in the table that follows.

```
<complexType name=" WSPendingApproval">
  <sequence>
    <element name="action" nillable="true" type="soapenc:string"/>
    <element name="reason" nillable="true" type="soapenc:string"/>
    <element name="workflowId" nillable="true" type="soapenc:int"/>
  </sequence>
</complexType>
```

Element	Description	Modify?	Return Code	Fault String
	Only approvers can use this CLI	No	-99	Access denied
	This API is applicable only when Resource record workflow is turned on.	No	-276	Applicable only when Resource record workflow is turned on
action	Specify "Approve" or "Reject".	No	-161	Input action null! Invalid action: *action*

Element	Description	Modify?	Return Code	Fault String
reason	The text entered at creation for the allocation reason.	No.		
workflowId	The id of the pending approval request retrieved using an Export*Item*PendingApproval, for example, ExportResourceRecordPendingApproval.	No	-162	Workflow id not found: *id*

RestoreDeletedDevice

Overview

The **RestoreDeletedDevice** API enables the web service client to restore deleted devices.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **RestoreDeletedDevice** request and response messages.

```
<wsdl:message name="restoreDeletedDeviceRequest">
   <wsdl:part name="inpDevice" type="tns2:WSRestoreDevice"/>
</wsdl:message>
<wsdl:message name="restoreDeletedDeviceResponse">
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSRestoreDevice**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSRestoreDevice

Below is the portion of *Imports.wsdl* that describes **WSRestoreDevice**, the parameter structure passed to **restoreDeletedDevice**. The elements are described in the table that follows.

```
<complexType name="WSRestoreDevice">
    <sequence>
     <element name="MACAddress" nillable="true" type="soapenc:string"/>
     <element name="addressType" nillable="true" type="soapenc:string"/>
     <element name="adminLoginId" nillable="true" type="soapenc:string"/>
     <element name="container" nillable="true" type="soapenc:string"/>
     <element name="description" nillable="true" type="soapenc:string"/>
     <element name="deviceType" nillable="true" type="soapenc:string"/>
     <element name="domainName" nillable="true" type="soapenc:string"/>
     <element name="domainType" nillable="true" type="soapenc:string"/>
     <element name="duid" nillable="true" type="soapenc:string"/>
     <element name="hostname" nillable="true" type="soapenc:string"/>
     <element name="hwType" nillable="true" type="soapenc:string"/>
     <element name="ignoreDuplicateWarning" nillable="true" type="soapenc:string"/>
     <element name="ipAddress" nillable="true" type="soapenc:string"/>
     <element name="restoreId" nillable="true" type="soapenc:int"/>
     <element name="blockName" nillable="true" type="soapenc:string"/>
     <element name="dateTime" nillable="true" type="soapenc:string"/>
    </sequence>
</complexType>
```

Element	Description	Required	Return Code	Fault String
MACAddress	The Hardware(Mac)Address of the deleted device.	No	-119	Hardware Address *MACAddress* already in use
addressType	Address type of the deleted device.	No		
adminLoginId	The login id of the administrator who deleted the record.	No		
container	Name of the Container from which the device was deleted.	No.		
description	Description of the deleted device.	No		
deviceType	Device type of the deleted device	No		
domainName	Domain name of the deleted device.	No		
domainType	Domain type of the deleted device.	No		
Duid	DHCP Unique Identifier	No		
Hostname	The hostname of the deleted device.	No	-80	Found duplicate hostname for *hostname*
hwType		No		
ignoreDuplicateWarning	This field is used by RestoreDeletedDevice API. Export puts a false by default for this. Change this to True before using this output for restoring to ignore Duplicate Warnings for mac address or hostname.	No		
ipAddress	The ipaddress of the deleted device.	No	-269	IP Address i*paddress* already in use
restoreId	The id used internally to identify the record. It is required to restore this record via RestoreDeletedDevice.	Yes	-306 -108	No restore record found with the given id. Restore ID is required
blockName	The name of the block that the device was deleted from.	No		
dateTime	The date and time when the device was deleted.	No		

Other Return Codes and Fault Strings

Return Code	Fault String
-2	Restore Failed

RestoreDeletedResourceRecord

Overview

The **RestoreDeletedResourceRecord** API enables the web service client to restore deleted resource records.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **RestoreDeletedResourceRecord** request and response messages.

```
<wsdl:message name="restoreDeletedResourceRecordRequest">
   <wsdl:part name="inpRR" type="tns2:WSRestoreResourceRecord"/>
</wsdl:message>
<wsdl:message name="restoreDeletedResourceRecordResponse">
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSRestoreResourceRecord**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSRestoreResourceRecord

Below is the portion of *Imports.wsdl* that describes **WSRestoreResourceRecord**, the parameter structure passed to **restoreDeletedResourceRecord**. The elements are described in the table that follows.

```
<complexType name=" WSRestoreResourceRecord">
  <sequence>
    <element name="TTL" nillable="true" type="soapenc:string"/>
    <element name="admin" nillable="true" type="soapenc:string"/>
    <element name="comment" nillable="true" type="soapenc:int"/>
     <element name="data" nillable="true" type="soapenc:string"/>
    <element name="dateTime" nillable="true" type="soapenc:string"/>
    <element name="domain" nillable="true" type="soapenc:string"/>
    <element name="domainType" nillable="true" type="soapenc:string"/>
    <element name="hostname" nillable="true" type="soapenc:string"/>
    <element name="ipAddress" nillable="true" type="soapenc:string"/>
    <element name="owner" nillable="true" type="soapenc:string"/>
    <element name="resourceRecClass" nillable="true" type="soapenc:string"/>
    <element name="resourceRecType" nillable="true" type="soapenc:string"/>
    <element name="restoreId" nillable="true" type="soapenc:int"/>
    <element name="ignoreDuplicateOwnerWarning" nillable="true" type="soapenc:string"/>
  </sequence>
</complexType>
```

Element	Description	Return Code	Fault String
TTL	The Time To Live for the record.		
admin	The login id of the administrator who deleted the record.		
comment	Comment text associated with the resource record.		
data	The data portion of the resource record. The format is dependent on the type specified.	-224	RDATA field of resource record must be less than 512 characters
dateTime	The date and time when the record was deleted.		
domain	Domain name of resource record.	-230	Admin does not have write access for the domain
domainType	Domain type of resource record.		
hostname	The device host name.		
ipAddress	The IP Address of the Device.		
owner	The owner field of the resource record.	-136	Found duplicate resource record for owner
resourceRecClass	The Class of the Resource Record. Defaults to "IN".		
resourceRecordType	The Type of the resource Record.	-99	Access denied for *rr type*
restoreId	The ID used to identify the deleted record.	-306 -108	No restore record found with the given id. Restore ID is required
ignoreDuplicateOwnerWarning	True/false flag to indicate if restore should ignore a duplicate owner warning.		

Other Return Codes and Fault Strings

Return Code	Fault String
-2	Restore Failed

SplitBlock

Overview

The **SplitBlock** API enables the web service client to split an existing block into smaller blocks.

Request and Response Messages

Below is the portion of *Imports.wsdl* that describes the **splitBlock** request and response messages.

```
<wsdl:message name="splitBlockRequest">
  <wsdl:part name="blockName" type="soapenc:string"/>
  <wsdl:part name="container" type="soapenc:string"/>
  <wsdl:part name="targetStartAddress" type="soapenc:string"/>
  <wsdl:part name="targetSize" type="xsd:int"/>
  <wsdl:part name="equalSizes" type="xsd:boolean"/>
</wsdl:message>
<wsdl:message name="splitBlockResponse">
</wsdl:message>
```

Response

There is no data in the response.

Request Parameters

blockName

Specify the name of the block, for example, `10.0.0.0/24`. The system searches for non-free blocks by this name.

container

The container name must be specified if the block name is not unique, due to overlapping space or naming conventions.

targetStartAddress

Specify the start address of the target block. This is useful for creating a block using the specified start address and target block size. If no start address is specified, the start address of the block being split will be used.

targetSize

Specify the desired CIDR block size after the split. This parameter works in conjunction with the **equalSizes** parameter.

equalSizes

If true, the block is split such that all resulting blocks have the **targetSize** CIDR size. If false, the block is split such that the fewest number of new blocks is created, along with two blocks of **targetSize**.

For example, if a /24 is split with a target size of /27 with **`equalSizes`** set to true, the result will be eight /27 blocks. If a /24 is split with a target size of /27 with **`equalSizes`** set to false, the result will be one /25, one /26 and two /27 blocks.

Return Codes and Fault Strings

Return Code	Fault String
-36	Block not found
-97	Block not unique
-15	Invalid Target Block Size
-17	Invalid Block Status
-12	Invalid Block Address
-9	Subnet creation error
-104	Address pool cleanup error
-99	Access Denied

Gets

Overview

This section explains the web services available for retrieving individual objects from IPAM. These services can be used in conjunction with Imports to modify IPAM objects. Each of these services is available as an operation in the Gets web service. You can see the complete WSDL at: http://localhost:8080/inc-ws/services/Gets?wsdl

getAddressPool

This operation retrieves information about an address pool from IPAM.

This call can be used in conjunction with the **`ImportAddressPool`** call to modify an existing address pool. Use this call to retrieve an address pool based on its starting address. Modify the returned structure as needed and pass the modified structure to **`ImportAddressPool`**. **`ImportAddressPool`** will perform an update instead of a create due to the presence of an ID element in the address pool structure.

Request and Response Messages

Below is the portion of *Gets.wsdl* that describes the **`getAddressPool`** request and response messages.

```
<wsdl:message name="getAddressPoolResponse">
  <wsdl:part name="getAddressPoolReturn" type="tns2:WSAddrpool"/>
</wsdl:message>
<wsdl:message name="getAddressPoolRequest">
  <wsdl:part name="startAddress" type="soapenc:string"/>
  <wsdl:part name="container" type="soapenc:string"/>
</wsdl:message>
```

Response

If the address pool is found, a **`WSAddrpool`** structure is filled in with its information. This structure is also used by the **`ImportAddressPool`** API call.

Request

There are two parameters in the request.

Parameter	Description	Required
startAddr	The starting address of the Address Pool	Yes
Container	The container holding the block in which the address pool resides.	Only if startAddr is not unique due to overlapping blocks.

Return Codes

If the address pool is not found, the call will fail with an error code of -117.

getAdmin

This operation retrieves information about an administrator from IPAM.

This call can be used in conjunction with the **`ImportAdmin`** call to modify an existing administrator. Use this call to retrieve an administrator based on its Login ID. Modify the returned structure as needed and pass the modified structure to **`ImportAdmin`**. **`ImportAdmin`** will perform an update instead of an import due to the presence of an **id** element in the **`WSAdmin`** structure

Request and Response Messages

Below is the portion of *Gets.wsdl* that describes the **`getAdmin`** request and response messages.

```
<wsdl:message name="getAdminRequest">
      <wsdl:part name="adminName" type="soapenc:string"/>
</wsdl:message>
<wsdl:message name="getAdminResponse">
      <wsdl:part name="getAdminReturn" type="tns2:WSAdmin"/>
</wsdl:message>
```

Response

If the administrator is found, a **`WSAdmin`** structure is filled in with its information. See the **`ImportAdmin`** API call for more information about the elements in the wsdl.

Request

This request has one parameter.

Parameter Name	Description	Required
adminName	The Login ID of the administrator	Yes

Return Codes

Condition	Return Code
Login ID is required for getAdmin	-293
No admin found for login ID: *name*	-295

getAdminRole

This operation retrieves information about an administrator role from IPAM.

This call can be used in conjunction with the **ImportAdminRole** call to modify an existing administrator role. Use this call to retrieve an administrator role based on its role name. Modify the returned structure as needed and pass the modified structure to **ImportAdminRole**. **ImportAdminRole** will perform an update instead of an import due to the presence of an **id** element in the **WSAdminRole** structure.

Request and Response Messages

Below is the portion of *Gets.wsdl* that describes the **getAdminRole** request and response messages.

```
<wsdl:message name="getAdminRoleRequest">
      <wsdl:part name="adminRoleName" type="soapenc:string"/>
</wsdl:message>
<wsdl:message name="getAdminRoleResponse">
      <wsdl:part name="getAdminRoleReturn" type="tns2:WSAdminRole"/>
</wsdl:message>
```

Response

If the block is found, a **WSAdminRole** structure is filled in with its information. See the **ImportAdminRole** API call for more information about the elements in the wsdl.

Request

This request has one parameter.

Parameter Name	Description	Required
adminRoleName	The name of the administrator role	Yes

Return Codes

Condition	Return Code
Role name is required for get	-300
No admin role found for name: *name*	-303

getBlock

There are two calls to retrieve blocks:

- **getBlockByName**
- **getBlockByIpAddress**

Both calls return a **WSGenericBlock** data structure. (See the **ModifyBlock** call for a description of **WSGenericBlock**.) As their names suggest, these calls differ in how the block is located.

Use one of the above calls in conjunction with the **ModifyBlock** call to update an existing block. Modify the returned structure as needed and pass the modified structure to **ModifyBlock**. Note that the complete container path will be returned in the **WSGenericBlock**.

Request and Response Messages

Below is the portion of *Gets.wsdl* that describes the **getBlockByName** and **getBlockByIpAddress** request and response messages.

```
<wsdl:message name="getBlockByNameRequest">
  <wsdl:part name="name" type="soapenc:string"/>
  <wsdl:part name="container" type="soapenc:string"/>
</wsdl:message>
<wsdl:message name="getBlockByNameResponse">
  <wsdl:part name="getBlockByNameReturn" type="tns2:WSGenericBlock"/>
</wsdl:message>

<wsdl:message name="getBlockByIpAddressRequest">
  <wsdl:part name="ipAddress" type="soapenc:string"/>
  <wsdl:part name="container" type="soapenc:string"/>
  <wsdl:part name="size" type="xsd:int"/>
  <wsdl:part name="status" type="soapenc:string"/>
</wsdl:message>
<wsdl:message name="getBlockByIpAddressResponse">
  <wsdl:part name="getBlockByIpAddressReturn" type="tns2:WSGenericBlock"/>
</wsdl:message>
```

Response

If the block is found, a **WSGenericBlock** structure is filled in with its information. See the **ModifyBlock** API call for more information.

getBlockByName Request

This request has two parameters.

Parameter Name	Description	Required
Name	The Name of the block, e.g. 10.0.0.0/24	Yes
Container	The container holding the block.	Only if name is not unique.

getBlockByIpAddress Request

This request has four parameters.

Parameter Name	Description	Required
ipAddress	The starting IP Address of the block	Yes
container	The container holding the block	Only if the IP Address is not unique due to overlapping space.
size	The block's CIDR size.	Only if there are multiple blocks with the same starting address, e.g. 10.0.0.0/8 (aggregate) and 10.0.0.0/24 (child block).
status	The block's status. Valid values are "Free", "Reserved", "Aggregate", "Deployed", "FullyAssigned"	Only if there are multiple blocks with the same starting address. This can occur for aggregates where there is a free block with the same starting address and size.

Return Codes

Condition	Return Code
Container Not Found	-5
Block not Unique	-97
Invalid IP Address	-26
Block not Found	-36
Invalid Block Status	-17

getContainer

There is one call to retrieve a container: **getContainerByName**.

This call returns a **WSContainer** data structure.

Use this call in conjunction with the **ImportContainer** call to modify an existing container. Modify the returned structure as needed and pass the modified structure to **ImportContainer**. **ImportContainer** will update the container based on the presence of an ID element in the **WSContainer** structure.

Request and Response Messages

Below is the portion of *Gets.wsdl* that describes the **getContainerBy** request and response messages.

```
<wsdl:message name="getContainerByNameRequest">
  <wsdl:part name="containerName" type="soapenc:string"/>
</wsdl:message>
<wsdl:message name="getContainerByNameResponse">
  <wsdl:part name="getContainerByNameReturn" type="tns1:WSContainer"/>
</wsdl:message>
```

Response

If the Container is found, a **WSContainer** structure is filled in with its information. This structure is also used by the **ImportContainer** API call.

getContainerByName Request

There is one parameter in the request.

Parameter Name	Description	Required
containerName	The name of the container.	Yes

Return Codes

Condition	Return Code
Container not found	-5

getContainerParentHierarchy

This call returns a String[ ] data structure containing the fully qualified names of all parents of the named container.

Use this call to discover identically named logical containers the reside in different branches of your network structure or all the locations that an individual device container is referenced.

Request and Response Messages

Below is the portion of *Gets.wsdl* that describes the **`getContainerParentHierarchy`** request and response messages.

```
<wsdl:message name="getContainerParentHierarchyRequest">
  <wsdl:part name="containerName" type="soapenc:string"/>
</wsdl:message>
<wsdl:message name="getContainerParentHierarchyResponse">
  <wsdl:part name="getContainerParentHierarchyReturn" type="impl:ArrayOf_soapenc_string"/>
</wsdl:message>
```

Response

If the Container is found, a String[ ] structure is filled in with its parent hierarchy information where each entry represents the fully qualified path to the named container. For example suppose we search for device container "router" which happens to be attached to two locations in the network hierarchy. In this case the output of the API call might look like the following:

InControl/usa/state/pa/gov/dmv/router

InControl/usa/state/pa/gov/epa/router

getContainerParentHierarchy

There is one parameter in the request.

Parameter Name	Description	Required
containerName	The name of the container.	Yes

Return Codes

Condition	Return Code
Container not found	-5
Container name required	-42
SQL Exception	-53
Access denied	-99

getDevice

There are three calls to retrieve a device:

- **getDeviceByIPAddr**
- **getDeviceByMACAddress**
- **getDeviceByHostname**

All three calls return a **WSDevice** data structure. As their names suggest, they differ in how the device is located.

Use any of the three calls in conjunction with the **ImportDevice** call to modify an existing device. Modify the returned structure as needed and pass the modified structure to **ImportDevice**. **ImportDevice** will update the device based on the presence of an ID element in the **WSDevice** structure.

Request and Response Messages

Below is the portion of *Gets.wsdl* that describes the **getDeviceBy** request and response messages.

```
<wsdl:message name="getDeviceByIPAddrRequest">
  <wsdl:part name="ipAddress" type="soapenc:string"/>
  <wsdl:part name="container" type="soapenc:string"/>
</wsdl:message>
<wsdl:message name="getDeviceByIPAddrResponse">
  <wsdl:part name="getDeviceByIPAddrReturn" type="tns2:WSDevice"/>
</wsdl:message>

<wsdl:message name="getDeviceByHostnameRequest">
  <wsdl:part name="hostname" type="soapenc:string"/>
</wsdl:message>
<wsdl:message name="getDeviceByHostnameResponse">
  <wsdl:part name="getDeviceByHostnameReturn" type="tns2:WSDevice"/>
</wsdl:message>

<wsdl:message name="getDeviceByMACAddressRequest">
  <wsdl:part name="macAddress" type="soapenc:string"/>
</wsdl:message>
<wsdl:message name="getDeviceByMACAddressResponse">
  <wsdl:part name="getDeviceByMACAddressReturn" type="tns2:WSDevice"/>
</wsdl:message>
```

Response

If the Device is found, a **WSDevice** structure is filled in with its information. This structure is also used by the **ImportDevice** API call.

getDeviceByIPAddr Request

There are two parameters in the request.

Parameter Name	Description	Required
ipAddress	The IP address of the device	Yes
container	The container holding the block in which the address pool resides.	Only if ipAddress is not unique due to overlapping blocks.

getDeviceByMACAddress Request

There is one parameter in the request.

Parameter Name	Description	Required
macAddress	The MAC address of the device	Yes

getDeviceByHostname Request

There is one parameter in the request.

Parameter Name	Description	Required
hostname	The hostname address of the device	Yes

Return Codes

Condition	Return Code
Device Not Found	-120
Multiple IP Addresses found	-28
Invalid IP Address	-26
Invalid MAC Address	-122
Multiple Devices Found	-121
Missing Hostname	-80

getDeviceResourceRec

This operation retrieves information about a DNS resource record from IPAM.

This call can be used in conjunction with the **ImportDeviceResourceRecord** call to modify an existing DNS resource record. Use this call to retrieve a resource record based on its device. Modify the returned structure as needed and pass the modified structure to **ImportDeviceResourceRecord**. **ImportDeviceResourceRecord** will perform an update instead of a create due to the presence of an ID element in the resource record structure.

Request and Response Messages

Below is the portion of *Gets.wsdl* that describes the **getDeviceResourceRec** request and response messages.

```
<wsdl:message name="getDeviceResourceRecResponse">
  <wsdl:part name="getDeviceResourceRecReturn" type="tns2:WSDeviceResourceRec"/>
</wsdl:message>
<wsdl:message name=" getDeviceResourceRecRequest">
  <wsdl:part name="domainName" type="soapenc:string"/>
  <wsdl:part name="domainTypeName" type="soapenc:string"/>
  <wsdl:part name="owner" type="soapenc:string"/>
  <wsdl:part name="type" type="soapenc:string"/>
  <wsdl:part name="classtype" type="soapenc:string"/>
  <wsdl:part name="rdata" type="soapenc:string"/>
  <wsdl:part name="hostname" type="soapenc:string"/>
  <wsdl:part name="ipAddress" type="soapenc:string"/>
  <wsdl:part name="container" type="soapenc:string"/>
</wsdl:message>
```

Response

If the resource record is found, a **WSDeviceResourceRec** structure is filled in with its information. This structure is also used by the **ImportDeviceResourceRecord** API call.

Request

These are the request parameters:

Parameter Name	Description	Required
domainName	The name of the domain	Yes
domainNameType	The name of the domain type to which the domain belongs. Defaults to "Default".	No
owner	The OWNER section of the resource record.	No
type	The type of resource record.	Yes
classtype	The value currently supported is IN.	No
rdata	The text for the data area of the record.	No
hostname	The device host name.	Yes, unless IpAddress is specified.
ipAddress	The IP address of the device.	Yes, unless hostname is specified.
container	The name of the container that holds the device.	Yes, if ipAddress is in overlapping space.

Return Codes

Condition	Return Code
Exception reading from db	-2
Invalid IP Address	-26
IP Address not unique	-28
Domain not found	-60
No device with ipaddress or hostname	-120
Hostname not unique	-121
DNS resource record not found	-124
DNS resource record not unique	-125

getDhcpServer

There are two calls for retrieving information about a DHCP server:

- **getDhcpServerByName**
- **getDhcpServerByIpAddress**

Both return a **WSDhcpServer** structure. As their names suggest, they differ in how the server is located.

Use either of the calls in conjunction with the **ImportDhcpServer** call to modify an existing DHCP Server. Modify the returned structure as needed and pass the modified structure to **ImportDhcpServer**. **ImportDhcpServer** will update the server based on the presence of an ID element in the **WSDhcpServer** structure.

Request and Response Messages

Below is the portion of *Gets.wsdl* that describes the Dhcp Server request and response messages.

```
<wsdl:message name="getDhcpServerByNameRequest">
  <wsdl:part name="name" type="soapenc:string"/>
</wsdl:message>
<wsdl:message name="getDhcpServerByNameResponse">
  <wsdl:part name="getDhcpServerByNameReturn" type="tns2:WSDhcpServer"/>
</wsdl:message>

<wsdl:message name="getDhcpServerByIpAddressRequest">
  <wsdl:part name="ipAddress" type="soapenc:string"/>
</wsdl:message>
<wsdl:message name="getDhcpServerByIpAddressResponse">
  <wsdl:part name="getDhcpServerByIpAddressReturn" type="tns2:WSDhcpServer"/>
</wsdl:message>
Response
```

If the DHCP server is found, a **WSDhcpServer** structure is filled in with its information. This structure is also used by the **ImportDhcpServer** API call.

getDhcpServerByName Request

There is one parameter in the request.

Parameter Name	Description	Required
Name	The name of the DHCP Server	Yes

getDhcpServerByIpAddress Request

There is one parameter in the request.

Parameter Name	Description	Required
ipAddress	The IP Address of the DHCP Server	Yes

Return Codes

Condition	Return Code
DHCP Server not found	-94
Invalid IP Address	-26

getDomainResourceRec

This operation retrieves information about a DNS resource record from IPAM. This call can be used in conjunction with the **ImportDomainResourceRecord** call to modify an existing DNS resource record. Use this call to retrieve a resource record that is not bound to a particular device. Modify the returned structure as needed and pass the modified structure to **ImportDomainResourceRecord**. **ImportDomainResourceRecord** will perform an update instead of a create due to the presence of an ID element in the resource record structure.

Request and Response Messages

Below is the portion of *Gets.wsdl* that describes the **getDomainResourceRec** request and response messages.

```
<wsdl:message name="getDomainResourceRecResponse">
  <wsdl:part name="getDomainResourceRecReturn" type="tns2:WSDomainResourceRec"/>
</wsdl:message>
<wsdl:message name=" getDomainResourceRecRequest">
  <wsdl:part name="domainName" type="soapenc:string"/>
  <wsdl:part name="domainTypeName" type="soapenc:string"/>
  <wsdl:part name="owner" type="soapenc:string"/>
  <wsdl:part name="type" type="soapenc:string"/>
  <wsdl:part name="classtype" type="soapenc:string"/>
  <wsdl:part name="rdata" type="soapenc:string"/>
</wsdl:message>
```

Response

If the resource record is found, a **WSDomainResourceRec** structure is filled in with its information. This structure is also used by the **ImportDomainResourceRecord** API call.

Request Parameters:

Parameter Name	Description	Required
domainName	The name of the domain	Yes
domainNameType	The name of the domain type to which the domain belongs. Defaults to "Default".	No
owner	The OWNER section of the resource record.	Yes
type	The type of resource record.	Yes
classtype	The value currently supported is IN.	No
rdata	The text for the data area of the record.	No

Return Codes

Condition	Return Code
Exception reading from db	-2
Domain not found	-60
Owner required	-89
DNS Resource Record not unique	-125
Domain name required	-135
DNS resource record not found	-136

getEffectiveDhcpServersForContainer

This call retrieves the effective DHCP servers for the named container. These are the DHCP servers that are available for selection when adding a block to the container.

If the named container has DHCP Servers attached to it and the admin has access to them, this call returns those. If the container doesn't have any servers attached directly to it, then it returns the list of accessible DHCP servers from the parent tree. If the container is multiparented (device container attached to two different containers) it gets the union of the DHCP servers from all parents. If the named container and its parents have no DHCP servers attached to them, this call returns all the DHCP servers that the admin has access to.

Request and Response Messages

Below is the portion of *Gets.wsdl* that describes the **`getEffectiveDhcpServersForContainer`** request and response messages.

```
<wsdl:message name="getEffectiveDhcpServersForContainerResponse">
       <wsdl:part name="getEffectiveDhcpServersForContainerReturn"
       type="impl:ArrayOf_tns2_WSDhcpServer"/>
</wsdl:message>


<wsdl:message name="getEffectiveDhcpServersForContainerResponse">

       <wsdl:part name="getEffectiveDhcpServersForContainerReturn"
       type="impl:ArrayOf_tns2_WSDhcpServer"/>

</wsdl:message>
```

Response

If the DHCPServers are found, a **`WSDhcpServer`** [ ] structure is filled in with the information.

getContainerParentHierarchy

There is one parameter in the request.

Parameter Name	Description	Required
containerName	The name of the container.	Yes

Return Codes

Condition	Return Code
Container not found	-5
Container name required	-42

getNetworkElement

There are two calls to retrieve a network element:

- **getNetworkElementByName**
- **getNetworkElementByNameOrIpAddress**

Both calls return a **WSNetworkElement** data structure. As their names suggest, they differ in how the network element is located.

Either call can be used in conjunction with the **ImportNetworkElement** call to modify an existing network element. Use a call to retrieve a network element based on its name and/or IP Address. Modify the returned structure as needed and pass the modified structure to **ImportNetworkElement**. **ImportNetworkElement** will perform an update instead of a create due to the presence of an **id** element in the **WSNetworkElement** structure.

Request and Response Messages

Below is the portion of *Gets.wsdl* that describes the **getNetworkElementBy** request and response messages.

```
<wsdl:message name="getNetworkElementByNameResponse">
      <wsdl:part name="getNetworkElementByNameReturn" type="tns2:WSNetworkElement"/>
</wsdl:message>

<wsdl:message name="getNetworkElementByNameRequest">
      <wsdl:part name="netElementName" type="soapenc:string"/>
</wsdl:message>

<wsdl:message name="getNetworkElementByNameOrIpAddressResponse">
      <wsdl:part name="getNetworkElementByNameOrIpAddressReturn"
type="tns2:WSNetworkElement"/>
</wsdl:message>

<wsdl:message name="getNetworkElementByNameOrIpAddressRequest">
      <wsdl:part name="netElementName" type="soapenc:string"/>
      <wsdl:part name="netElementIpAddress" type="soapenc:string"/>
</wsdl:message>
```

Response

If the block is found, a **WSNetworkElement** structure is filled in with its information. See the **ImportNetworkElement** API call for more information about the elements in the wsdl.

getNetworkElementByName Request

This request has one parameter.

Parameter Name	Description	Required
netElementName	The name of the network element	Yes

getNetworkElementByNameOrIpaddress Request

This request has two parameters.

Parameter Name	Description	Required
netElementName	The name of the network element	No
netElementIpAddress	The IP Address of the network element	Yes

Return Codes

Condition	Return Code
Specify network element name, not unique by ip address: *address*	-28
NetworkElement *name* not found	-33
Network Element Name is required	-35
Network Element IP Address is required	-127
Admin is not authorized to access Network Element functions	-318

getNetelementInterface

There is one call to retrieve a network element interface: **getNetElementInterface**.

This call returns a **WSNetElementInterface** data structure.

Use this call in conjunction with the **ImportNetElementInterface** call to modify an existing network element interface. Modify the returned structure as needed and pass the modified structure to **ImportNetElementInterface**.
ImportNetElementInterface updates the network element interface based on the presence of an ID element in the **WSNetElementInterface** structure.

Request and Response Messages

Below is the portion of *Gets.wsdl* that describes the **getNetElementInterface** request and response messages.

```
<wsdl:message name="getNetelementInterfaceRequest">
  <wsdl:part name="neName" type="soapenc:string"/>
  <wsdl:part name="iName" type="soapenc:string"/>
</wsdl:message>
<wsdl:message name="getNetelementInterfaceResponse">
  <wsdl:part name="getNetelementInterfaceReturn"  type="tns2:WSNetElementInterface"/>
 </wsdl:message>
```

Response

If the Network Element Interface is found, a **WSNetElementInterface** structure is filled in with its information. This structure is also used by the **ImportNetElementInterface** API call.

getNetelementInterface Request

There are two parameters in the request.

Parameter Name	Description	Required
neName	The name of the network element.	Yes
iName	The name of the interface	Yes

Return Codes

Condition	Return Code
Network element not found	-33
Interface not found	-19

getNetworkLink

This operation retrieves information about a network link from IPAM.

This call can be used in conjunction with the **ImportNetworkLink** call to modify an existing network link. Use this call to retrieve a network link based on its name. Modify the returned structure as needed and pass the modified structure to **ImportNetworkLink**. **ImportNetworkLink** will perform an update instead of a create due to the presence of an **id** element in the **WSNetworkLink** structure.

Request and Response Messages

Below is the portion of *Gets.wsdl* that describes the **getNetworkLinkByName** request and response messages.

```
<wsdl:message name="getNetworkLinkByNameResponse">
      <wsdl:part name="getNetworkLinkByNameReturn" type="tns2:WSNetworkLink"/>
</wsdl:message>

<wsdl:message name="getNetworkLinkByNameRequest">
      <wsdl:part name="netLinkName" type="soapenc:string"/>
</wsdl:message>
```

Response

If the block is found, a **WSNetworkLink** structure is filled in with its information. See the **ImportNetworkLink** API call for more information about the elements in the wsdl.

Request

This request has one parameter.

Parameter Name	Description	Required
Name	The Name of the network link	Yes

Return Codes

Condition	Return Code
NetworkLink *name* not found	-279
Admin is not authorized to access Network Link functions	-231

getPrefixPool

There are two calls for retrieving information about a Prefix Pool:

- **getPrefixPoolByName**
- **getPrefixPoolByStartAddr**

Both return a **WSPrefixPool** structure.

Use either of the calls in conjunction with the **ImportPrefixPool** call to modify an existing Prefix Pool. Modify the returned structure as needed and pass the modified structure to **ImportPrefixPool**. **ImportPrefixPool** will perform an update instead of a create due to the presence of an ID element in the prefix pool structure.

Request and Response Messages

Below is the portion of *Gets.wsdl* that describes the Prefix Pool request and response messages.

```
<wsdl:message name="getPrefixPoolByStartAddrResponse">
      <wsdl:part name="getPrefixPoolByStartAddrReturn" type="tns2:WSPrefixPool"/>
</wsdl:message>

<wsdl:message name="getPrefixPoolByStartAddrRequest">
     <wsdl:part name="addr" type="soapenc:string"/>
</wsdl:message>

<wsdl:message name="getPrefixPoolByNameResponse">
     <wsdl:part name="getPrefixPoolByNameReturn" type="tns2:WSPrefixPool"/>
</wsdl:message>

<wsdl:message name="getPrefixPoolByNameRequest">
     <wsdl:part name="name" type="soapenc:string"/>
</wsdl:message>
```

Response

If the prefix pool is found, a **WSPrefixPool** structure is filled in with its information. This structure is also used by the **ImportPrefixPool** API call.

getPrefixPoolByName Request

There is one parameter in the request.

Parameter Name	Description	Required
Name	The name of the Prefix Pool	Yes

getPrefixPoolByStartAddr Request

There is one parameter in the request.

Parameter Name	Description	Required
Addr	The Start Address of the Prefix Pool	Yes

Return Codes

If the prefix pool is not found, the call will fail with an error code of -262.

Tasks

Overview

This section explains the web services available for issuing tasks to IPAM, and for querying the status of IPAM tasks. Each of these services is available as an operation in the TaskInvocation web service. You can see the complete WSDL at: http://localhost:8080/inc-ws/services/TaskInvocation?wsdl

Return Codes

The following codes are returned as negative integers from those services returning type **int**. For more information, look for error messages in the web services log.

Code	Description
-1	System Error: Serious error preventing the operation from continuing, such as failure to connect to the database.
-2	Access Denied: User failed security check.
-3	Invalid Parameter: Missing or Invalid parameters related to a particular call
-4	Resource Not Found: Resource that service requires does not exist

ArpDiscoverNetElement

Overview

The **arpDiscoverNetElement** API enables the web service client to issue an immediate Arp Discovery task to collect host information from a router's ARP cache.

Request and Response Messages

Below is the portion of *TaskInvocation.wsdl* that describes the **arpDiscoverNetElement** request and response messages.

```
<wsdl:message name="arpDiscoverNetElementResponse">
   <wsdl:part name="arpDiscoverNetElementReturn" type="xsd:int"/>
</wsdl:message>
<wsdl:message name="arpDiscoverNetElementRequest">
   <wsdl:part name="netelement" type="soapenc:string"/>
   <wsdl:part name="performNetHostAdd" type="xsd:boolean"/>
   <wsdl:part name="updateReclaim" type="xsd:boolean"/>
   <wsdl:part name="ignoreDuplicates" type="xsd:boolean"/>
    <wsdl:part name="reverseLookup" type="xsd:boolean"/>
    <wsdl:part name="importInvalid" type="xsd:boolean"/>
</wsdl:message>
```

Response

If the task is scheduled successfully, the web service returns the task number. Pass this task number to the **taskStatus** service to obtain the status of that task. If the task is not scheduled successfully, the negative integer returned in the response contains a code as described in the chapter introduction.

Request

The parts passed as input from the client to the web service are described in the next section.

Parameters

Below is the portion of *TaskInvocation.wsdl* that describes the parameter structure passed to **arpDiscoverNetElement**. The individual parameters are described in the table that follows.

```
<wsdl:operation name="arpDiscoverNetElement"
          parameterOrder="netelement performNetHostAdd updateReclaim ignoreDuplicates
reverseLookup importInvalid">
 <wsdl:input message="impl:arpDiscoverNetElementRequest"
          name="arpDiscoverNetElementRequest"/>
 <wsdl:output message="impl:arpDiscoverNetElementResponse"
          name="arpDiscoverNetElementResponse"/>
</wsdl:operation>
```

Parameter name	Description and accepted values
netelement	Name or IP Address of the Net Element to run ARP discover on.
performNetHostAdd	Set to true to automatically add new hosts that are found during discovery.
updateReclaim	Set to true to update the last discovered counters for blocks and hosts defined within the network element.
ignoreDuplicates	Set to true to ignore duplicate hostnames encountered while adding new hosts.
reverseLookup	Set to true to do a reverse DNS lookup and attempt to resolve the FQDN for this address.
importInvalid	Set to true to import records the router has flagged as invalid.

DHCPConfigurationAllFiles

Overview

The **dhcpConfigurationAllFiles** API enables the web service client to issue an immediate DHCP Configuration task for a specified DHCP server.

Request and Response Messages

Below is the portion of *TaskInvocation.wsdl* that describes the **dhcpConfigurationAllFiles** request and response messages.

```
<wsdl:message name="dhcpConfigurationAllFilesRequest">
   <wsdl:part name="name" type="soapenc:string"/>
   <wsdl:part name="ip" type="soapenc:string"/>
   <wsdl:part name="stopOnError" type="xsd:boolean"/>
   <wsdl:part name="pushOnlyChanges" type="xsd:boolean"/>
   <wsdl:part name="updateFailovers" type="xsd:boolean"/>
</wsdl:message>
<wsdl:message name="dhcpConfigurationAllFilesResponse">
   <wsdl:part name="dhcpConfigurationAllFilesReturn" type="xsd:int"/>
</wsdl:message>
```

Response

If the task is scheduled successfully, the web service returns the task number. Pass this task number to the **taskStatus** service to obtain the status of that task. If the task is not scheduled successfully, the negative integer returned in the response contains a code as described in the chapter introduction.

Request

The parts passed as input from the client to the web service are described in the next section.

Parameters

Below is the portion of *TaskInvocation.wsdl* that describes the parameter structure passed to **dhcpConfigurationAllFiles**. The individual parameters are described in the table that follows.

```
      <wsdl:operation name="dhcpConfigurationAllFiles" parameterOrder="name ip stopOnError
pushOnlyChanges updateFailovers">
         <wsdl:input message="impl:dhcpConfigurationAllFilesRequest"
name="dhcpConfigurationAllFilesRequest"/>
         <wsdl:output message="impl:dhcpConfigurationAllFilesResponse"
name="dhcpConfigurationAllFilesResponse"/>
      </wsdl:operation>
```

Part name	Description
name	Name of the DHCP Server. Either this or the server IP must be specified.
IP	IP Address of the DHCP Server. Either this or the server name must be specified.
stopOnError	Specify whether to stop if an error is encountered or ignore the error and continue.
pushOnlyChanges	Specify whether to push the files even if there have been no configuration changes or not.
updateFailovers	Specify whether the push should also update failover servers.

DHCPUtilization

Overview

The **dhcpUtilization** API enables the web service client to issue an immediate DHCP Collection task to collect statistics on the utilization of a DHCP server.

Request and Response Messages

Below is the portion of *TaskInvocation.wsdl* that describes the **dhcpUtilization** request and response messages.

```
<wsdl:message name="dhcpUtilizationResponse">
  <wsdl:part name="dhcpUtilizationReturn" type="xsd:int" />
```

```
</wsdl:message>
</wsdl:message><wsdl:message name="dhcpUtilizationRequest">
  <wsdl:part name="adminId" type="soapenc:string" />
  <wsdl:part name="password" type="soapenc:string" />
  <wsdl:part name="elementName" type="soapenc:string" />
  <wsdl:part name="ipAddress" type="soapenc:string" />
</wsdl:message>
```

Response

If the task is scheduled successfully, the web service returns the task number. Pass this task number to the **taskStatus** service to obtain the status of that task. If the task is not scheduled successfully, the negative integer returned in the response contains a code as described in the chapter introduction.

Request

The parts passed as input from the client to the web service are described in the next section.

Parameters

Below is the portion of *TaskInvocation.wsdl* that describes the parameter structure passed to **dhcpUtilization**. The individual parameters are described in the table that follows.

```
<wsdl:operation name="dhcpUtilization"
  parameterOrder="adminId password elementName ipAddress">
  <wsdl:input message="impl:dhcpUtilizationRequest" name="dhcpUtilizationRequest" />
  <wsdl:output message="impl:dhcpUtilizationResponse" name="dhcpUtilizationResponse" />
</wsdl:operation>
```

Part name	Description and accepted values	Required
adminId	IPAM administrator's user id	Yes
password	IPAM administrator's password	Yes
elementName	Name of the DHCP server for which IPAM will collect statistics.	Yes, when IP Address or FQDN is not specified.
ipAddress	IP Address or fully-qualified (FQDN) of the DHCP server for which IPAM will collect statistics.	Yes, when the elementName is not specified.

DiscoverNetElement

Overview

The **discoverNetElement** API enables the web service client to issue an immediate Discover task to discover the interfaces bound to a network element already defined in IPAM.

Request and Response Messages

Below is the portion of *TaskInvocation.wsdl* that describes the **discoverNetElement** request and response messages.

```
<wsdl:message name="discoverNetElementResponse">
  <wsdl:part name="discoverNetElementReturn" type="xsd:int" />
</wsdl:message>
<wsdl:message name="discoverNetElementRequest">
  <wsdl:part name="adminId" type="soapenc:string" />
  <wsdl:part name="password" type="soapenc:string" />
  <wsdl:part name="elementName" type="soapenc:string" />
  <wsdl:part name="ipAddress" type="soapenc:string" />
</wsdl:message>
```

Response

If the task is scheduled successfully, the positive integer returned by the web service corresponds to the task number. That task number can then be passed to the TaskStatus service to obtain the status of that task. If the task is not scheduled successfully, the negative integer returned in the response contains a code as described in the chapter introduction.

Request

The parts passed as input from the client to the web service are described in the next section.

Parameters

Below is the portion of *TaskInvocation.wsdl* that describes the parameter structure passed to **discoverNetElement**. The individual parameters are described below.

```
<wsdl:operation name="discoverNetElement"
  parameterOrder="adminId password elementName ipAddress">
  <wsdl:input message="impl:discoverNetElementRequest" name="discoverNetElementRequest" />
  <wsdl:output message="impl:discoverNetElementResponse" name="discoverNetElementResponse" />
</wsdl:operation>
```

Part name	Description and accepted values	Required
adminId	IPAM administrator's user ID	Yes
password	IPAM administrator's password	Yes
elementName	Name of Network Element (device) to discover.	Yes, when IP Address or FQDN is not specified.
ipAddress	IP Address or fully-qualified (FQDN) of the device to discover.	Yes, when the elementName is not specified.

DNSConfigurationAllFiles

Overview

The **dnsConfigurationAllFiles** API enables the web service client to issue an immediate DNS all files push request (configuration and zones) for the specified DNS server.

Request and Response Messages

Below is the portion of *TaskInvocation.wsdl* that describes the **dnsConfigurationAllFiles** request and response messages.

```
<wsdl:message name=" dnsConfigurationAllFilesResponse">
  <wsdl:part name=" dnsConfigurationAllFilesReturn" type="xsd:int" />
</wsdl:message>
<wsdl:message name="dnsConfigurationAllFilesRequest">
  <wsdl:part name="name" type="soapenc:string" />
  <wsdl:part name="ip" type="soapenc:string" />
  <wsdl:part name="abortfailedcheck" type="xsd:boolean"/>
  <wsdl:part name="checkconf" type="xsd:boolean"/>
  <wsdl:part name="checkzones " type="xsd:boolean"/>
</wsdl:message>
```

Response

If the task is scheduled successfully, the web service returns the task number. Pass this task number to the **taskStatus** service to obtain the status of that task. If the task is not scheduled successfully, the negative integer returned in the response contains a code as described in the chapter introduction.

Request

The parts passed as input from the client to the web service are described in the next section.

Parameters

Below is the portion of *TaskInvocation.wsdl* that describes the parameter structure passed to **dnsConfigurationAllFiles**. The individual parameters are described in the table that follows.

```
<wsdl:operation name="dnsConfigurationAllFiles" parameterOrder="name ip abortfailedcheck
checkconf checkzones">
  <wsdl:input message="impl:dnsConfigurationAllFilesRequest"
name="dnsConfigurationAllFilesRequest"/>
  <wsdl:output message="impl:dnsConfigurationAllFilesResponse"
name="dnsConfigurationAllFilesResponse"/>
</wsdl:operation>
```

Part name	Description and accepted values	Required
name	Name of the DNS server.	name or ip must be specified.
ip	IP address of the DNS server.	name or ip must be specified.
abortfailedcheck	Set to true if the push should halt if either check fails (checkconf and checkzones).	yes

checkconf	If true, the configuration file check script specified when the DNS network service was created will be run against the named.conf file created by the DNS Configuration task.	yes
checkzones	If true, the zone file check script specified when the DNS network service was created will be run against the zone files created by the DNS Configuration task.	Yes

DNSConfigurationChangedZones

Overview

The **dnsConfigurationChangedZones** API enables the web service client to issue an immediate DNS push, creating the configuration file and only the changed zone files for the specified DNS server.

Request and Response Messages

Below is the portion of *TaskInvocation.wsdl* that describes the **dnsConfigurationChangedZones** request and response messages.

```
<wsdl:message name=" dnsConfigurationChangedZonesResponse">
  <wsdl:part name="dnsConfigurationChangedZonesReturn" type="xsd:int" />
</wsdl:message>
<wsdl:message name="dnsConfigurationChangedZonesRequest ">
  <wsdl:part name="name" type="soapenc:string" />
  <wsdl:part name="ip" type="soapenc:string" />
  <wsdl:part name="abortfailedcheck" type="xsd:boolean"/>
  <wsdl:part name="checkzones " type="xsd:boolean"/>
</wsdl:message>
```

Response

If the task is scheduled successfully, the web service returns the task number. Pass this task number to the **taskStatus** service to obtain the status of that task. If the task is not scheduled successfully, the negative integer returned in the response contains a code as described in the chapter introduction.

Request

The parts passed as input from the client to the web service are described in the next section.

Parameters

Below is the portion of *TaskInvocation.wsdl* that describes the parameter structure passed to **dnsConfigurationChangedZones**. The individual parameters are described in the table that follows.

```
<wsdl:operation name="dnsConfigurationChangedZones" parameterOrder="name ip
abortfailedcheck checkzones">
  <wsdl:input message="impl:dnsConfigurationChangedZonesRequest"
name="dnsConfigurationChangedZonesRequest"/>
  <wsdl:output message="impl:dnsConfigurationChangedZonesResponse"
name="dnsConfigurationChangedZonesResponse"/>
</wsdl:operation>
```

Part name	Description and accepted values	Required
name	Name of the DNS server.	name or ip must be specified.
ip	IP address of the DNS server.	name or ip must be specified.
abortfailedcheck	Set to true if the push should halt if the checkzones option is specified and the check fails.	yes

checkzones	If true, the zone file check script specified when the DNS network service was created will be run against the zone files created by the DNS Configuration task.	yes

DNSConfigurationSelectedZones

Overview

The **dnsConfigurationSelectedZones**API enables the web service client to issue an immediate DNS push, creating the configuration file and zone files for the specified DNS server and selected zones.

Request and Response Messages

Below is the portion of *TaskInvocation.wsdl* that describes the **dnsConfigurationSelectedZones** request and response messages.

```
<wsdl:message name=" dnsConfigurationSelectedZonesResponse">
  <wsdl:part name="dnsConfigurationSelectedZonesReturn" type="xsd:int" />
</wsdl:message>
<wsdl:message name="dnsConfigurationSelectedZonesRequest ">
  <wsdl:part name="name" type="soapenc:string"/>
  <wsdl:part name="ip" type="soapenc:string"/>
  <wsdl:part name="view" type="soapenc:string"/>
  <wsdl:part name="zone" type="soapenc:string"/>
  <wsdl:part name="abortfailedcheck" type="xsd:boolean"/>
  <wsdl:part name="checkzones " type="xsd:boolean"/>
</wsdl:message>
```

Response

If the task is scheduled successfully, the web service returns the task number. Pass this task number to the **taskStatus** service to obtain the status of that task. If the task is not scheduled successfully, the negative integer returned in the response contains a code as described in the chapter introduction.

Request

The parts passed as input from the client to the web service are described in the next section.

Parameters

Below is the portion of *TaskInvocation.wsdl* that describes the parameter structure passed to **dnsConfigurationSelectedZones**. The individual parameters are described in the table that follows.

```
<wsdl:operation name="dnsConfigurationSelectedZones" parameterOrder="name ip view zone
abortfailedcheck checkzones">
  <wsdl:input message="impl:dnsConfigurationSelectedZonesRequest"
name="dnsConfigurationSelectedZonesRequest"/>
  <wsdl:output message="impl:dnsConfigurationSelectedZonesResponse"
name="dnsConfigurationSelectedZonesResponse"/>
</wsdl:operation>
```

Part name	Description and accepted values	Required
name	Name of the DNS server.	name or ip must be specified.
ip	IP address of the DNS server.	name or ip must be specified.

view	View name, defaults to "Default"	no
zone	Zone name to push. If null or "all", all zones are pushed	no
abortfailedcheck	Set to true if the push should halt if the checkzones option is specified and the check fails.	yes
checkzones	If true, the zone file check script specified when the DNS network service was created will be run against the zone files created by the DNS Configuration task.	yes

DNSDDNSAllRRs

Overview

The **dnsDDNSAllRRs** API enables the web service client to issue an immediate request to send all resource records for the selected zone, or all zones, for the specified DNS server via RFC2136 dynamic DNS updates.

Request and Response Messages

Below is the portion of *TaskInvocation.wsdl* that describes the **dnsDDNSAllRRs** request and response messages.

```
<wsdl:message name=" dnsDDNSAllRRsResponse">
  <wsdl:part name=" dnsDDNSAllRRsReturn" type="xsd:int" />
</wsdl:message>
<wsdl:message name=" dnsDDNSAllRRsRequest">
  <wsdl:part name="name" type="soapenc:string"/>
  <wsdl:part name="ip" type="soapenc:string"/>
  <wsdl:part name="view" type="soapenc:string"/>
  <wsdl:part name="zone" type="soapenc:string"/>
  <wsdl:part name=" userCreatedOption" type="xsd:boolean"/>
</wsdl:message>
```

Response

If the task is scheduled successfully, the web service returns the task number. Pass this task number to the **taskStatus** service to obtain the status of that task. If the task is not scheduled successfully, the negative integer returned in the response contains a code as described in the chapter introduction.

Request

The parts passed as input from the client to the web service are described in the next section.

Parameters

Below is the portion of *TaskInvocation.wsdl* that describes the parameter structure passed to **dnsDDNSAllRRs**. The individual parameters are described in the table that follows.

```
<wsdl:operation name="dnsDDNSAllRRs" parameterOrder="name ip view zone userCreatedOption">
  <wsdl:input message="impl:dnsDDNSAllRRsRequest" name="dnsDDNSAllRRsRequest"/>
  <wsdl:output message="impl:dnsDDNSAllRRsResponse" name="dnsDDNSAllRRsResponse"/>
</wsdl:operation>
```

Part name	Description and accepted values	Required
name	Name of the DNS server.	name or ip must be specified.
ip	IP address of the DNS server.	name or ip must be specified.
view	View name, defaults to “Default”	no
zone	Zone name to push. If null or “all”, all zones are pushed	no
userCreatedOption	When set to true, sends only resource records created in IPAM. Use this option to periodically refresh the records in Microsoft AD DNS to prevent their scavenging, while not interfering with the intended scavenging of dynamic records.	yes

DNSDDNSChangedRRs

Overview

The **dnsDDNSChangedRRs** API enables the web service client to issue an immediate request to send changed resource records for the specified DNS server via RFC2136 dynamic DNS updates.

Request and Response Messages

Below is the portion of *TaskInvocation.wsdl* that describes the **dnsDDNSChangedRRs** request and response messages.

```
<wsdl:message name=" dnsDDNSChangedRRsResponse">
  <wsdl:part name=" dnsDDNSChangedRRsReturn" type="xsd:int" />
</wsdl:message>
<wsdl:message name=" dnsDDNSChangedRRsRequest">
  <wsdl:part name="name" type="soapenc:string"/>
  <wsdl:part name="ip" type="soapenc:string"/>
</wsdl:message>
```

Response

If the task is scheduled successfully, the web service returns the task number. Pass this task number to the **taskStatus** service to obtain the status of that task. If the task is not scheduled successfully, the negative integer returned in the response contains a code as described in the chapter introduction.

Request

The parts passed as input from the client to the web service are described in the next section.

Parameters

Below is the portion of *TaskInvocation.wsdl* that describes the parameter structure passed to **dnsDDNSChangedRRs**. The individual parameters are described in the table that follows.

```
<wsdl:operation name="dnsDDNSChangedRRs" parameterOrder="name ip">
  <wsdl:input message="impl:dnsDDNSChangedRRsRequest" name="dnsDDNSChangedRRsRequest"/>
  <wsdl:output message="impl:dnsDDNSChangedRRsResponse" name="dnsDDNSChangedRRsResponse"/>
</wsdl:operation>
```

Part name	Description and accepted values	Required
name	Name of the DNS server.	name or ip must be specified.
ip	IP address of the DNS server.	name or ip must be specified.

GetTask

Overview

The **getTask** API enables the web service client to query the status of tasks and receive detailed information about those tasks.

Request and Response Messages

Below is the portion of *TaskInvocation.wsdl* that describes the **getTask** request and response messages.

```
<wsdl:message name="getTaskRequest">
  <wsdl:part name="taskId" type="xsd:int" />
</wsdl:message>
<wsdl:message name="getTaskResponse">
  <wsdl:part name="getTaskReturn" type="impl:ArrayOf_soapenc_string" />
</wsdl:message>
```

Response

GetTask will return the status of the queried task as a string array with the following information:

Element	Description
0	Task ID
1	Service
2	Scope
3	Status
4	Process start time

Request

The input from the client to the web service is described in the next section.

Parameters

Below is the portion of *TaskInvocation.wsdl* that describes the parameter structure passed to **getTask** . The parameter is described in the table that follows.

```
<wsdl:operation name="getTask" parameterOrder="taskId">
  <wsdl:input message="impl:getTaskRequest" name="getTaskRequest" />
  <wsdl:output message="impl:getTaskResponse" name="getTaskResponse"/>
</wsdl:operation>
```

Parameter name	Description and accepted values	Required
taskId	The task number to query.	Yes

GetTaskStatus

Overview

The **getTaskStatus** API enables the web service client to query the status of tasks.

Request and Response Messages

Below is the portion of *TaskInvocation.wsdl* that describes the **getTaskStatus** request and response messages.

```
<wsdl:message name="getTaskStatusResponse">
  <wsdl:part name="getTaskStatusReturn" type="soapenc:string" />
</wsdl:message>
<wsdl:message name="getTaskStatusRequest">
  <wsdl:part name="taskId" type="xsd:int" />
</wsdl:message>
```

Response

GetTaskStatus returns the status of the queried task as one of the following strings:

- NOTSTARTED
- QUEUED
- INPROGRESS
- COMPLETE
- COMPLETEWITHERRORS
- ERROR

For more detailed information about the task, use the **getTask** service, described next in this section.

Request

The input from the client to the web service is described in the next section.

Parameters

Below is the portion of *TaskInvocation.wsdl* that describes the parameter structure passed to **getTaskStatus**. The parameter is described below.

```
<wsdl:operation name="getTaskStatus" parameterOrder="taskId">
  <wsdl:input message="impl:getTaskStatusRequest" name="getTaskStatusRequest" />
  <wsdl:output message="impl:getTaskStatusResponse" name="getTaskStatusResponse" />
</wsdl:operation>
```

Part name	Description and accepted values	Required
taskId	The task number to query.	Yes

GlobalNetElementSync

Overview

The **globalNetElementSync** API enables the web service client to issue an immediate Global Synchronization task for all network elements in IPAM that are flagged for inclusion in the Global Sync process.

Request and Response Messages

Below is the portion of *TaskInvocation.wsdl* that describes the **globalNetElementSync** request and response messages.

```
<wsdl:message name="globalNetElementSyncResponse">
  <wsdl:part name="globalNetElementSyncReturn" type="xsd:int" />
</wsdl:message>
<wsdl:message name="globalNetElementSyncRequest">
  <wsdl:part name="adminId" type="soapenc:string" />
  <wsdl:part name="password" type="soapenc:string" />
</wsdl:message>
```

Response

If the task is scheduled successfully, the positive integer returned by the web service will correspond to the task number. That task number can then be passed to the **taskStatus** service to obtain the status of that task. If the task is not scheduled successfully, the negative integer returned in the response contains a code as described in the chapter introduction.

Request

The parts passed as input from the client to the web service are described in the next section.

Parameters

Below is the portion of *TaskInvocation.wsdl* that describes the parameter structure passed to **globalNetElementSync** . The individual parameters are described below.

```
<wsdl:operation name="globalNetElementSync" parameterOrder="adminId password">
  <wsdl:input message="impl:globalNetElementSyncRequest"
    name="globalNetElementSyncRequest" />
  <wsdl:output message="impl:globalNetElementSyncResponse"
    name="globalNetElementSyncResponse" />
</wsdl:operation>
```

Part name	Description and accepted values	Required
adminId	IPAM administrator's user id	Yes
password	IPAM administrator's password	Yes

GlobalNetServiceSync

Overview

The **globalNetServiceSync** API enables the web service client to issue an immediate Global Synchronization task for all network services in IPAM that are flagged for inclusion in the Global Sync process.

Request and Response Messages

Below is the portion of *TaskInvocation.wsdl* that describes the **globalNetServiceSync** request and response messages.

```
<wsdl:message name="globalNetServiceSyncResponse">
  <wsdl:part name="globalNetServiceSyncReturn" type="xsd:int" />
</wsdl:message
  <wsdl:message name="globalNetServicetSyncRequest">
  <wsdl:part name="adminId" type="soapenc:string" />
  <wsdl:part name="password" type="soapenc:string" />
</wsdl:message>
```

Response

If the task is scheduled successfully, the positive integer returned by the web service will correspond to the task number. That task number can then be passed to the **taskStatus** service to obtain the status of that task. If the task is not scheduled successfully, the negative integer returned in the response contains a code as described in the chapter introduction.

Request

The parts passed as input from the client to the web service are described in the next section.

Parameters

Below is the portion of *TaskInvocation.wsdl* that describes the parameter structure passed to **globalNetServiceSync** . The individual parameters are described below.

```
<wsdl:operation name="globalNetServiceSync" parameterOrder="adminId password">
  <wsdl:input message="impl:globalNetServiceSyncRequest"
    name="globalNetServiceSyncRequest" />
  <wsdl:output message="impl:globalNetServiceSyncResponse"
    name="globalNetServiceSyncResponse" />
</wsdl:operation>
```

Part name	Description and accepted values	Required
adminId	IPAM administrator's user id	Yes
password	IPAM administrator's password	Yes

GlobalRollup

Overview

The **globalRollup** API enables the web service client to issue an immediate Global Rollup task to collect statistics and perform regression analysis.

Request and Response Messages

Below is the portion of *TaskInvocation.wsdl* that describes the **globalRollup** request and response messages.

```
<wsdl:message name="globalRollupResponse">
  <wsdl:part name="globalRollupReturn" type="xsd:int" />
</wsdl:message>
<wsdl:message name="globalRollupRequest">
  <wsdl:part name="adminId" type="soapenc:string" />
  <wsdl:part name="password" type="soapenc:string" />
  <wsdl:part name="periodLength" type="xsd:int" />
  <wsdl:part name="_periodType" type="soapenc:string" />
</wsdl:message>
```

Response

If the task is scheduled successfully, the positive integer returned by the web service will correspond to the task number. That task number can then be passed to the **taskStatus** service to obtain the status of that task. If the task is not scheduled successfully, the negative integer returned in the response contains a code as described in the chapter introduction.

Request

The parts passed as input from the client to the web service are described in the next section.

Parameters

Below is the portion of *TaskInvocation.wsdl* that describes the parameter structure passed to **globalRollup** . The individual parameters are described in the table that follows.

```
<wsdl:operation name="globalRollup"
  parameterOrder="adminId password periodLength _periodType">
  <wsdl:input message="impl:globalRollupRequest" name="globalRollupRequest" />
  <wsdl:output message="impl:globalRollupResponse" name="globalRollupResponse" />
</wsdl:operation>
```

Part name	Description and accepted values	Required
adminId	IPAM administrator's user id	Yes
password	IPAM administrator's password	Yes
periodLength	The number of time periods to be included in the regression. If not specified, defaults to value set in System Policies.	Yes
_periodType	The type of time period. Accepted values are: **D** (days), **W** (weeks), **M** (months), and **Y** (years).	Yes

Exports

Overview

This section explains the web services available for retrieving information from IPAM. Each of these services is available as an operation in the Exports web service. You can see the complete WSDL at:

http://localhost:8080/inc-ws/services/Exports?wsdl

Export Categories

There are two categories of Export web services. The first category consists of legacy APIs that were available in the initial version of IPAM. The second category consists of a newer set of APIs that provide a more flexible request and response format.

Legacy Web Services

The legacy web service APIs are designed to accept one or more request parameters which define the filter used to export objects from the IPAM database. In addition, these legacy APIs return a string response which contains a list of objects. Each object's fields are comma-delimited, and each object in the list is separated by a newline character. The legacy web service APIs are the following:

- **`ExportNetElementsAsCSV`**
- **`ExportAllNetElementsAsCSV`**
- **`ExportNetServicesAsCSV`**
- **`ExportAllNetServicesAsCSV`**

Next Generation Web Services

The new web service APIs are designed to accept a string request which contains a query defining the filter used to export objects from the IPAM database. These new APIs return a structure which represents the object being exported. The format of the structure is such that it can be directly used for the corresponding import web service. The new web service APIs are the following:

- **ExportAdmin**
- **ExportAdminRole**
- **ExportContainer**
- **ExportRootBlock**
- **ExportChildBlock**
- **ExportDevice**
- **ExportDeviceResourceRecord**
- **ExportDeviceRestoreList**
- **ExportDomainResourceRecord**
- **ExportNetworkElement**
- **ExportNetworkLink**
- **ExportPrefixPool**
- **ExportResourceRecordPendingApproval**
- **ExportResourceRecordPendingApprovalStatus**
- **ExportResourceRecordRestoreList**

Legacy Web Services

ExportNetElementsAsCSV

Overview

The **exportNetElementsAsCSV** API enables the web service client to issue a request to retrieve a list of Network Elements from IPAM. This service enables the client to filter the list of Network Elements retrieved.

Request and Response Messages

Below is the portion of *Exports.wsdl* that describes the **exportNetElementsAsCSV** request and response messages.

```
<wsdl:message name="exportNetElementsAsCSVResponse">
  <wsdl:part name="exportNetElementsAsCSVReturn" type="soapenc:string" />
</wsdl:message>
<wsdl:message name="exportNetElementsAsCSVRequest">
  <wsdl:part name="elementName" type="soapenc:string" />
  <wsdl:part name="vendorName" type="soapenc:string" />
  <wsdl:part name="modelDesc" type="soapenc:string" />
  <wsdl:part name="ipaddr" type="soapenc:string" />
  <wsdl:part name="globalsync" type="soapenc:string" />
  <wsdl:part name="agentName" type="soapenc:string" />
  <wsdl:part name="elementType" type="soapenc:string" />
</wsdl:message>
```

Response

The string that is returned contains the list of Network Elements matching the selection criteria specified in the request. Each Network Element description is separated by a new line character. The values within each Network Element description are separated by commas, and described in the table below. Fields that are not required may not always contain values. Fields H, I, J and K will not be exported, since these could contain sensitive information. The columns are preserved to maintain conformity with the **ImportNetElement** API.

Col	Field	Description	Required
A	Name	The name of the Network Element. This can be any combination of letters and numbers.	Yes
B	IP Address/FQDN	The IP address or fully-qualified domain name (FQDN) of the Network Element. This is a valid IPv4 or IPv6 IP address, or a fully-qualified host name.	Yes
C	Vendor	The vendor of the Network Element. Vendor is predefined in IPAM.	Yes when Model is specified.
D	Model	The model name of the Network Element. Model is predefined in IPAM.	Yes when Vendor is specified.
E	Type	The type of Network Element. Accepted values are **CMTS**, **Router**, **Switch**, or **VPN**.	Yes
F	Global Sync	Whether or not to include this Network Element in the Global Sync process. Value is **true** or **false**.	Yes

Col	Field	Description	Required
G	Agent Name	The exact name of the IPAM Agent that is responsible for contacting this Network Service.	Yes
H	Telnet User	A user name used to telnet to this device.	No
I	Telnet Password	A password used by the telnet user to telnet to this device.	No
J	Enable Password	Password used to enter "enabled" or "privileged" mode on the device.	No
K	Read Community String	The community string used by SNMP to read details from this network element.	No
L	Interfaces	A list of enabled interfaces. Multiple interfaces are specified by separating each interface with the '|' character.	No
M	V3 Username	Required if using SNMP V3.	No
N	V3 Authentication Protocol	Either **MD5** or **SHA1**. Leave blank or set to **NONE** if no authentication.	No
O	V3 Authentication Password	Required if field N is set to either **MD5** or **SHA1**	No
P	V3 Privacy Protocol	Only **DES** supported at this time. Leave blank or set to **NONE** if no privacy.	No
Q	V3 Privacy Password	Required if field P is set to **DES**.	No
R	V3 Context Name	SNMP V3 Context name, if needed.	No
S	V3 Engine ID	SNMP V3 Engine ID, if needed.	No

Request

The parts passed as input from the client to the web service are described in the next section. They are used to filter the information retrieved from IPAM. None are required. To retrieve all Network Elements, use **`ExportAllNetElementsAsCSV`**.

Parameters

Below is the portion of *Exports.wsdl* that describes the parameter structure passed to **`exportNetElementsAsCSV`** . The individual parameters are described in the table that follows. Note that none of the parameters are required, since they are used as a filter for the information retrieved.

```
<wsdl:operation name="exportNetElementsAsCSV"
  parameterOrder="elementName vendorName modelDesc ipaddr globalsync agentName
  elementType">
  <wsdl:input message="impl:exportNetElementsAsCSVRequest"
    name="exportNetElementsAsCSVRequest" />
  <wsdl:output message="impl:exportNetElementsAsCSVResponse"
    name="exportNetElementsAsCSVResponse" />
</wsdl:operation>
```

Part name	Description and accepted values
elementName	The name of the Network Element.
vendorName	The vendor of the Network Element.
modelDesc	The model name of the Network Element.
ipAddress	IP Address or fully-qualified (FQDN) of the Network Element.

Part name	Description and accepted values
globalsync	Whether or not to include this Network Element in the Global Synch process. Specify **Y** (yes) or **N** (no).
agentName	The exact name of the IPAM Agent that is responsible for contacting this Network Element.
elementType	The type of Network Element. Specify **CMTS**, **Router, Switch** or **VPN.**

ExportAllNetElementsAsCSV

Overview

The **exportAllNetElementsAsCSV** API enables the web service client to issue a request to retrieve a list of all of the Network Elements from IPAM. To filter the request, use the **exportNetElementsAsCSV** service.

Request and Response Messages

Below is the portion of *Exports.wsdl* that describes the **exportAllNetElementsAsCSV** request and response messages.

```
<wsdl:message name="exportAllNetElementsAsCSVResponse">
  <wsdl:part name="exportAllNetElementsAsCSVReturn" type="soapenc:string" />
</wsdl:message>
<wsdl:message name="exportAllNetElementsAsCSVRequest" />
```

Response

The string that is returned contains the list of all of the Network Elements. See the Response section of **ExportNetElementsAsCSV** for format and details.

Request

There are no input parameters for this web service.

Parameters

Below is the portion of Exports.wsdl that describes the parameter structure passed to **exportAllNetElementsAsCSV** .

```
<wsdl:operation name="exportAllNetElementsAsCSV">
  <wsdl:input message="impl:exportAllNetElementsAsCSVRequest"
    name="exportAllNetElementsAsCSVRequest" />
  <wsdl:output message="impl:exportAllNetElementsAsCSVResponse"
    name="exportAllNetElementsAsCSVResponse" />
</wsdl:operation>
```

ExportNetServicesAsCSV

Overview

The **exportNetServicesAsCSV** API enables the web service client to issue a request to retrieve a list of DHCP Network Services from IPAM. This API enables the web service client to filter the list of Network Services retrieved.

Request and Response Messages

Below is the portion of *Exports.wsdl* that describes the **exportNetServicesAsCSV** request and response messages.

```
<wsdl:message name="exportNetServicesAsCSVResponse">
  <wsdl:part name="exportNetServicesAsCSVReturn" type="soapenc:string" />
</wsdl:message>
<wsdl:message name="exportNetServicesAsCSVRequest">
  <wsdl:part name="serviceName" type="soapenc:string" />
  <wsdl:part name="vendorName" type="soapenc:string" />
  <wsdl:part name="containerName" type="soapenc:string" />
  <wsdl:part name="ipaddr" type="soapenc:string" />
  <wsdl:part name="globalsync" type="soapenc:string" />
  <wsdl:part name="agentName" type="soapenc:string" />
  <wsdl:part name="serviceType" type="soapenc:string" />
</wsdl:message>
```

Response

The string that is returned contains the list of Network Services matching the selection criteria specified in the request. Each Network Service description is separated by a new line character. The values within each Network Service description are separated by commas, and described in the table below. Fields that are not required may not always contain values. Fields H and I will not be exported, since these could contain sensitive information. The columns are preserved to maintain conformity with the **ImportNetService** API.

Col	Field	Accepted Values	Required
A	Name	The name of the Network Service. This can be any combination of letters and numbers.	Yes
B	IP Address/FQDN	The IP address or fully-qualified domain name (FQDN) of the Network Service. This must be a valid IPv4 or IPv6 IP address, or a fully-qualified host name.	Yes
C	Type	The type of Network Service. This is always **dhcp**.	No
D	Product name	The Network Service product name. This is a value already defined in IPAM, for example, **CNR DHCP**.	Yes
E	Agent name	The name of the IPAM Agent that is responsible for contacting this Network Service.	Yes
F	Global Sync	Whether or not this Network Service is included in the Global Sync process. Values are **true** or **false**.	Yes
G	Collection Method	The method by which the IPAM Agent collects data from the Network Service. Values are **scp** or **ftp**.	Yes

Col	Field	Accepted Values	Required
H	User name for collection	The username used by the collection method (scp or ftp) to log in to the remote server. This is exported as "username".	Yes
I	Password for collection	The password used by the collection method (scp or ftp) to log in to the remote server. Used in conjunction with the 'User name for collection'. This is exported as "password".	Yes
J	Collection port	The port number the collection method (scp or ftp) is listening on.	No
K	Container(s)	No longer used.	
L	VendorInfo	Vendor specific information for the product's collection type. Each item of information is specified in this single field by separating each field with the '|' character. For collection types **qip**,**adc**, **msft** and **isc**, the information includes the DHCP Configuration file pathname and DHCP Active Lease file pathname. For example, /opt/qip/dhcp/dhcpd.conf |/opt/qip/dhcp/dhcp.db or c:\qip\dhcp\dhcpd.conf |c:\qip\dhcp\dhcp.db For collection type **cnr**, the information includes the Path/Executable of NRCD command, the NRCMD user id, the NRCMD password and the Cluster Name. For example, /opt/cnr/bin/nrcmd|myuserid|mypass| cluster1	No

Request

The parts passed as input from the client to the web service are described in the next section. They are used to filter the information retrieved from IPAM. None are required. To retrieve all Network Services, use **`ExportAllNetServicesAsCSV`**.

Parameters

Below is the portion of *Exports.wsdl* that describes the parameter structure passed to **`exportNetServicesAsCSV`** . The individual parameters are described in the table that follows. Note that none of the parameters are required, since they are used as a filter for the information retrieved.

```
<wsdl:operation name="exportNetServicesAsCSV" parameterOrder="serviceName vendorName
  containerName ipaddr globalsync agentName serviceType">
  <wsdl:input message="impl:exportNetServicesAsCSVRequest"
    name="exportNetServicesAsCSVRequest" />
  <wsdl:output message="impl:exportNetServicesAsCSVResponse"
    name="exportNetServicesAsCSVResponse" />
</wsdl:operation>
```

Part name	Description and accepted values
serviceName	The name of the Network Service.
vendorName	The Network Service product name.
containerName	The container the service is attached to.
ipAddress	IP Address or fully-qualified (FQDN) of the Network Service.
globalsync	Whether or not this Network Service is included in the Global Synch process. Specify **Y** (yes) or **N** (no).
agentName	The exact name of the IPAM Agent that is responsible for contacting this Network Service.
serviceType	The type of Network Service. Specify **dhcp**.

ExportAllNetServicesAsCSV

Overview

The **exportAllNetServicesAsCSV** API enables the web service client to issue a request to retrieve a list of all of the DHCP Network Services from IPAM. To filter the request, use the **exportNetServicesAsCSV** API.

Request and Response Messages

Below is the portion of *Exports.wsdl* that describes the **exportAllNetServicesAsCSV** request and response messages.

```
<wsdl:message name="exportAllNetServicesAsCSVResponse">
  <wsdl:part name="exportAllNetServicesAsCSVReturn" type="soapenc:string" />
</wsdl:message>
<wsdl:message name="exportAllNetServicesAsCSVRequest" />
```

Response

The string that is returned contains the list of all of the Network Services. See the Response section of **ExportNetServicesAsCSV** on page 380 for format and details.

Request

There are no input parameters for this web service.

Parameters

Below is the portion of Exports.wsdl that describes the parameter structure passed to **exportAllNetServicesAsCSV**.

```
<wsdl:operation name="exportAllNetSevicesAsCSV">
  <wsdl:input message="impl:exportAllNetSevicesAsCSVRequest"
    name="exportAllNetSevicesAsCSVRequest" />
  <wsdl:output message="impl:exportAllNetSevicesAsCSVResponse"
    name="exportAllNetSevicesAsCSVResponse" />
</wsdl:operation>
```

Next Generation Web Services

Selectors

The new web services APIs accept a single string parameter for the request. This request string specifies a query which is used to filter the list of exported objects. The query syntax consists of one or more selectors, combined into a Boolean expression. For example, to export the Device with a hostname of "mydevice", the request query string would be as follows:

```
name='mydevice'
```

Selectors which are based on a text field can support the keywords "begins", "ends", or "contains" to support wildcarding. For example, to export all Devices with a hostname beginning with "my", the request query string would be as follows:

```
name begins 'my'
```

The export filter can be further refined by combining additional selectors using the Boolean operators "and" and "or". For example, to export all Devices with a hostname beginning with "my" and with a device type of "PC", the request query string would be as follows:

```
name begins 'my' and devicetype='PC'
```

Use parentheses to apply specific precedence in expressions that utilize multiple Boolean operators.

Each new export web service supports a specific set of selectors. Please refer to each API definition on the following pages for the supported selector syntax.

Options

Some of the new web services APIs also support a second parameter, which is used to pass options to the service. Refer to the WSDL and the sections that follow for more information.

Paging

The new web services APIs also support the concept of *paging* through the export results. Some queries may result in thousands of exported objects. Due to memory and network constraints, it is not feasible to return all the results in a single response. Therefore, the web services client should specify the starting point within the list of results, as well as the number of results to return. This is accomplished using the **`WSContext`** object.

Each new web service *must* be initialized by the client by calling the initialization method associated with the export web service. Therefore, the client will always perform at least two web service requests to export objects from IPAM. For example, when exporting devices, the client must call **`initExportDevice`** first, followed by a call to **`exportDevice`**. The export service initialization APIs take the query string as the request, and return a **`WSContext`** object in the response. The export services APIs themselves take the **`WSContext`** object as the request, and return an array of exported objects in the response.

Sessions

Initialization calls are linked to subsequent export calls by using sessions. The initialization call creates a session and returns a session identifier as part of the SOAP envelope. This session identifier must be provided on all subsequent export calls, or an error occurs.

If you are using the Java Axis package to generate your web services client, configure your client to use the **`SimpleSessionHandler`**, as described in the documentation. If not, the details of the session handling follow.

The session identifier is returned as part of the SOAP Header. The namespace is `http://xml.apache.org/axis/session`. The element name is `sessionID`. An example of the returned header follows, where the value of the `sessionID` is 12345678.

```
<soapenv:Header>
  <ns1:sessionID soapenv:mustUnderstand="0" xmlns:ns1="http://xml.apache.org/axis/session">
    12345678
  </ns1:sessionID>
</soapenv:Header>
```

The response processing for the `init*` calls must capture this element and value. The subsequent `export*` calls must include this element and value in the SOAP Header. Without this, the `export*` calls cannot correlate with the init call, and return an error.

WSContext

Below is the portion of *Exports.wsdl* that describes **`WSContext`**, the parameter structure returned by the `initExport*` APIs, and passed to the `export*` APIs. The elements are described in the table that follows.

```
<complexType name="WSContext">
  <sequence>
    <element name="contextId" nillable="true" type="soapenc:string" />
    <element name="contextType" nillable="true" type="soapenc:string" />
    <element name="filter" nillable="true" type="soapenc:string" />
    <element name="firstResultPos" type="xsd:int" />
    <element name="maxResults" type="xsd:int" />
    <element name="options" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="query" nillable="true" type="soapenc:string" />
    <element name="resultCount" type="xsd:int" />
    <element name="internalResultCount" type="xsd:int" />
  </sequence>
</complexType>
```

Element	Description
contextId	Reserved
contextType	Reserved
filter	Reserved
firstResultPos	Reserved
maxResults	The number of result records to export.
options	Reserved
query	Reserved
resultCount	Reserved
internalResultCount	Reserved

ExportAdmin

Overview

The **ExportAdmin** API enables the web service client to issue a request to retrieve a list of administrators from IPAM. This service enables the client to filter the list of administrators retrieved.

Initialization

Before the **ExportAdmin** API is called, the web service client *must* call **initExportAdmin** to initialize the API. The portion of *Exports.wsdl* that describes the **initExportAdmin** request and response messages is shown below.

```
<wsdl:message name="initExportAdminResponse">
      <wsdl:part name="initExportAdminReturn" type="tns2:WSContext"/>
</wsdl:message>
<wsdl:message name="initExportAdminRequest">
      <wsdl:part name="in0" type="soapenc:string"/>
      <wsdl:part name="in1" type="impl:ArrayOf_soapenc_string"/>
</wsdl:message>
```

Request

The query string is passed as input from the client to the **initExportAdmin** web service. The query string syntax is defined previously. Supported selectors for exporting administrators are defined in the following table.

Selector	Description	Example
loginId	The login ID of the Admin to export. Partial names are supported using the begins/ends/contains qualifiers.	loginId='HRAdmin123' loginId begins 'HR' loginId ends '23' loginId contains 'adm'
role	The role of the Admins to export. Partial names are supported using the begins/ends/contains qualifiers.	role ='My Admin' role begins 'My' role ends 'Admin' role contains 'ad'
type	The administrator type to export. Specify **MASTER**|**NORMAL**|**READONLY**	type=NORMAL
lastName	The last name of the Admins to export. Partial names are supported using the begins/ends/contains qualifiers.	lastName="Smith" lastName begins "Sm" lastName ends "th" lastName contains "mi"
firstName	The first name of the Admins to export. Partial names are supported using the begins/ends/contains qualifiers.	firstName="John" firstName begins "J" firstName ends "hn" firstName contains "oh"

Response

The response from the **initExportAdmin** web service is a **WSContext** object defined previously, and *must* be included in each successive call to **ExportAdmin**, as described below.

Service Invocation

Below is the portion of *Exports.wsdl* that describes the **ExportAdmin** request and response messages.

```
<wsdl:message name="ExportAdminRequest">
      <wsdl:part name="in0" type="tns2:WSContext"/>
</wsdl:message>
<wsdl:message name="ExportAdminResponse">
      <wsdl:part name="ExportAdminReturn" type="impl:ArrayOf_tns2_WSAdmin"/>
</wsdl:message>
```

Request

The **WSContext** passed as input by the client web service is the **WSContext** object returned by the **initExportAdmin** service defined above, and has the **maxResults** field set to a default value of 100. When this context is provided to a subsequent call to **ExportAdmin**, the number of exported administrators is limited to the first 100 that match the criteria in the given query filter. The web service client may change this **maxResults** attribute of the **WSContext** before any call to the **ExportAdmin** service to modify the size of the resultant **WSAdmin** object array. However, the value specified by the client cannot exceed 100.

Response

The result returned from the **ExportAdmin** service is an array of **WSAdmin** objects matching the selection criteria specified in the query filter. The **WSAdmin** can then be modified and/or imported using the **importAdmin** API. The format of the **WSAdmin** matches that defined by the **importAdmin**.

WSAdmin

For a complete description of **WSAdmin**, please refer to the ImportAdmin API section of this guide.

ExportAdminRole

Overview

The **ExportAdminRole** API enables the web service client to issue a request to retrieve a list of administrator roles from IPAM. This service enables the client to filter the list of administrator roles retrieved.

Initialization

Before the **ExportAdminRole** API is called, the web service client *must* call **initExportAdminRole** to initialize the API. The portion of *Exports.wsdl* that describes the **initExportAdminRole** request and response messages is shown below.

```
<wsdl:message name="initExportAdminRoleResponse">
      <wsdl:part name="initExportAdminRoleReturn" type="tns2:WSContext"/>
</wsdl:message>
<wsdl:message name="initExportAdminRoleRequest">
      <wsdl:part name="in0" type="soapenc:string"/>
      <wsdl:part name="in1" type="impl:ArrayOf_soapenc_string"/>
</wsdl:message>
```

Request

The query string is passed as input from the client to the **initExportAdminRole** web service. The query string syntax is defined previously. Supported selectors for exporting administrators are defined in the following table.

Selector	Description	Example
name	The name of the Admin role to export. Partial names are supported using the begins/ends/contains qualifiers.	name='HR role' name begins 'HR name ends 'role name contains 'ro'
description	The description of the administrator role to export. Partial descriptions are supported using the begins/ends/contains qualifiers.	description contains 'something'

Response

The response from the **initExportAdminRole** web service is a **WSContext** object defined previously, and *must* be included in each successive call to **ExportAdminRole**, as described below.

Service Invocation

Below is the portion of *Exports.wsdl* that describes the **ExportAdminRole** request and response messages.

```
<wsdl:message name="ExportAdminRoleRequest">
      <wsdl:part name="in0" type="tns2:WSContext"/>
</wsdl:message>
<wsdl:message name="ExportAdminRoleResponse">
      <wsdl:part name="ExportAdminRoleReturn" type="impl:ArrayOf_tns2_WSAdminRole"/>
</wsdl:message>
```

Request

The **WSContext** passed as input by the client web service is the **WSContext** object returned by the **initExportAdminRole** service defined above, and has the **maxResults** field set to a default value of 100. When this context is provided to a subsequent call to **ExportAdminRole**, the number of exported administrator roles is limited to the first 100 that match the criteria in the given query filter. The web service client may change this **maxResults** attribute of the **WSContext** before any call to the **ExportAdminRole** service to modify the size of the resultant **WSAdminRole** object array. However, the value specified by the client cannot exceed 100.

Response

The result returned from the **ExportAdminRole** service is an array of **WSAdminRole** objects matching the selection criteria specified in the query filter. The **WSAdminRole** can then be modified and/or imported using the **importAdminRole** API. The format of the **WSAdminRole** matches that defined by the **importAdminRole**.

WSAdminRole

For a complete description of **WSAdminRole**, please refer to the ImportAdminRole API section of this guide.

ExportRootBlock

Overview

The **exportRootBlock** API enables the web service client to issue a request to retrieve a list of Root Blocks from IPAM. This service enables the client to filter the list of Root Blocks retrieved.

Initialization

Before the **exportRootBlock** API is called, the web service client *must* call **initExportRootBlock** to initialize the API. Below is the portion of *Exports.wsdl* that describes the **initExportRootBlock** request and response messages.

```
<wsdl:message name="initExportRootBlockRequest">
  <wsdl:part name="query" type="soapenc:string"/>
</wsdl:message>
<wsdl:message name="initExportRootBlockResponse">
  <wsdl:part name="initExportRootBlockReturn" type="tns2:WSContext"/>
</wsdl:message>
```

Request

The query string is passed as input from the client to the **initExportRootBlock** web service. The query string syntax is defined previously. Supported selectors for exporting root blocks are defined in the following table.

Selector	Description	Example
name	The name of the block to export. Partial names are supported using the begins/ends/contains qualifiers.	name='My Block' name begins 'My' name ends 'Block' name contains 'Bl'
block	The CIDR notation of the block to export. The accepted format for CIDR notation is 'block_address/block_size'.	block='10.0.0.0/24' block='10.0.*/24'
blocktype	The block type name of the block(s) to export.	blocktype='Private'
container	The container name of the block(s) to be exported.	container='Exton' container begins 'Ex' container ends 'ton' container contains 'xto'
ipaddress	An IP address that falls within the start and ending addresses of a block to be exported.	ipaddress='10.0.0.1'
ipaddressrange	A range of IP addresses that span one or more blocks' starting and ending addresses.	ipaddressrange='10.0.0.0-10.0.10.255'
udf	The field name/tag and value of a UDF attached to the block(s) to be exported.	UDF.myudf='myudf_value'

Response

The response from the **initExportRootBlock** web service is a **WSContext** object defined previously. This **WSContext** object *must* be included in each successive call to **exportRootBlock**, as described below.

Service Invocation

The portion of *Exports.wsdl* that describes the **exportRootBlock** request and response messages is shown below.

```
<wsdl:message name="exportRootBlockRequest">
  <wsdl:part name="context" type="tns2:WSContext" />
</wsdl:message>
<wsdl:message name="exportRootBlockResponse">
  <wsdl:part name="exportRootBlockReturn" type="impl:ArrayOf_tns1_WSRootBlock"/>
</wsdl:message>
```

Request

The **WSContext** passed as input by the client web service is the **WSContext** object returned by the **initExportRootBlock** service defined above. This **WSContext** has the **maxResults** field set to a default value of 100. When this context is provided to a subsequent call to **exportRootBlock**, the number of exported blocks is limited to the first 100 that match the criteria in the given query filter. The web service client may change the **maxResults** attribute of the **WSContext** before any call to the **exportRootBlock** service to modify the size of the resultant **WSRootBlock** object array. However, the value specified by the client cannot exceed 100.

Response

The result returned from the **exportRootBlock** service is an array of **WSRootBlock** objects matching the selection criteria specified in the query filter. The **WSRootBlocks** can then be modified and/or imported using the **importRootBlock** API. The format of the **WSRootBlock** matches that defined by the **importRootBlock**.

WSRootBlock

Below is the portion of *Exports.wsdl* that describes **WSRootBlock**, the array of structures returned by **exportRootBlock**. The elements are described in the table that follows.

```
<complexType name="WSRootBlock">
  <sequence>
    <element name="RIR" nillable="true" type="soapenc:string" />
    <element name="SWIPname" nillable="true" type="soapenc:string" />
    <element name="allocationReason" nillable="true" type="soapenc:string" />
    <element name="allocationReasonDescription" nillable="true" type="soapenc:string" />
    <element name="allowOverlappingSpace" nillable="true" type="soapenc:string" />
    <element name="blockAddr" nillable="true" type="soapenc:string" />
    <element name="blockName" nillable="true" type="soapenc:string" />
    <element name="blockSize" nillable="true" type="soapenc:string" />
    <element name="blockType" nillable="true" type="soapenc:string" />
    <element name="container" nillable="true" type="soapenc:string" />
    <element name="createReverseDomains" nillable="true" type="soapenc:string" />
    <element name="description" nillable="true" type="soapenc:string" />
    <element name="domainType" nillable="true" type="soapenc:string" />
    <element name="organizationId" nillable="true" type="soapenc:string" />
    <element name="userDefinedFields" nillable="true" type="impl:ArrayOf_soapenc_string" />
  </sequence>
```

```
</complexType>
```

Element	Accepted Values
RIR	The Regional Internet Registry this space was obtained from.
SWIPname	SWIP/Net name for the block.
allocationReason	The name of a pre-existing Allocation Reason.
allocationReasonDescription	A description of the reason for the allocation.
allowOverlappingSpace	Whether or not to allow duplicate (overlapping) address space in this block.
blockAddr	The starting address for the block.
blockName	A name for the block. Defaults to system supplied name of *Address space/Block size*.
blockSize	The size of the block in short-notation (e.g., 24 for a 255.255.255.0 network).
blockType	The Block Type of the block. If not specified, a block type of **Any** is assumed.
container	The name of the container holding the block.
createReverseDomains	Whether or not to automatically create reverse DNS domain(s) for this block. Accepted values are **true** or **false**. If not specified, defaults to **false**.
description	A description of the block.
domainType	The domain type of the reverse domain.
organizationId	The organization id for the Regional Internet Registry.
userDefinedFields	A string array containing one or more *name=value* pairs, where the *name* is the UDF field name/tag and the *value* is the desired value, for example**, State=PA**. If the UDF type is Checkbox, the valid values are **on** and **off**.

ExportChildBlock

Overview

The **exportChildBlock** API enables the web service client to issue a request to retrieve a list of Child Blocks from IPAM. This service enables the client to filter the list of Child Blocks retrieved.

Initialization

Before the **exportChildBlock** API is called, the web service client *must* call **initExportChildBlock** to initialize the API. The portion of *Exports.wsdl* that describes the **initExportChildBlock** request and response messages is shown below.

```
<wsdl:message name="initExportChildBlockRequest">
  <wsdl:part name="query" type="soapenc:string"/>
  <wsdl:part name="includeFreeBlocks" type="xsd:boolean"/>
</wsdl:message>
<wsdl:message name="initExportChildBlockResponse">
  <wsdl:part name="initExportChildBlockReturn" type="tns2:WSContext"/>
</wsdl:message>
```

Request

The query string is passed as input from the client to the **initExportChildBlock** web service. The query string syntax is defined previously. Supported selectors for exporting child blocks are defined in the following table.

In addition, the **initExportChildBlock** service accepts a Boolean flag that specifies if the free blocks maintained by IPAM should be included in the export.

Selector	Description	Example
name	The name of the block to export. Partial names are supported using the begins/ends/contains qualifiers.	name='My Block' name begins 'My' name ends 'Block' name contains 'Bl'
block	The CIDR notation of the block to export. The accepted format for CIDR notation is 'block_address/block_size'.	block='10.0.0.0/24' block='10.0.*/24'
blocktype	The block type name of the block(s) to export.	blocktype='Private'
container	The container name of the block(s) to be exported.	container='Exton' container begins 'Ex' container ends 'ton' container contains 'xto'
parentBlock	The name of the parent block of the block(s) to be exported. The special ^*container*= declarative allows the container name to be specified. This can be fully qualified or not.	parentName='172.16.0.0/23' parentName='172.16.0.0/23^container=North' parentName='172.16.0.0/23^container=/InControl/Canada/North'

Selector	Description	Example
Parent Container	Only applied when parentBlock is supplied. Specifies the name of the parent block's container. Useful in order to eliminate ambiguity by specifying the container name, fully qualified or not.	parentContainer='North' parentContainer='/InControl/Canada/North'
recursive	Valid options are: **true** and **container**. The **true** option is only applied when the parentBlock selector is also specified. When set to **true**, recursively exports all child blocks of the specified parent name. The **container** option is only applied when the container selector is also specified. When set to **container**, all child blocks from the named container branch of the container tree and all its descendants will be exported.	recursive=true recursive=container
status	The status of the block(s) to be exported. Valid options are: **free**, **aggregate**, **reserved**, **subnet**, **fullyassigned**.	status=aggregate
interface	The name of an interface to which the block is attached.	interface='eth0'
ipaddress	An IP address that falls within the start and ending addresses of a block to be exported.	ipaddress='10.0.0.1'
ipaddressrange	A range of IP addresses that span one or more blocks' starting and ending addresses.	ipaddressrange='10.0.0.0-10.0.10.255'
ipversion	The IP version of the block(s) to be exported. Valid options are: **v4** and **v6**.	ipversion=v4
udf	The name and value of a UDF attached to the block(s) to be exported.	UDF.myudf='myudf_value'

Response

The response from the **`initExportChildBlock`** web service is a **`WSContext`** object defined previously, and *must* be included in each successive call to **`exportChildBlock`**, as described below.

Service Invocation

Below is the portion of *Exports.wsdl* that describes the **`exportChildBlock`** request and response messages.

```
<wsdl:message name="exportChildBlockRequest">
  <wsdl:part name="context" type="tns2:WSContext" />
</wsdl:message>
<wsdl:message name="exportChildBlockResponse">
  <wsdl:part name="exportChildBlockReturn" type="impl:ArrayOf_tns1_WSChildSubnetBlock"/>
</wsdl:message>
```

Request

The **`WSContext`** passed as input by the client web service is the **`WSContext`** object returned by the **`initExportChildBlock`** service defined above, and has the **`maxResults`** field set

to a default value of 100. When this context is provided to a subsequent call to **exportChildBlock**, the number of exported blocks is limited to the first 100 that match the criteria in the given query filter. The web service client may change this **maxResults** attribute of the **WSContext** before any call to the **exportChildBlock** service to modify the size of the resultant **WSChildBlock** object array. However, the value specified by the client cannot exceed 100.

Response

The result returned from the **exportChildBlock** service is an array of **WSChildSubnetBlock** objects matching the selection criteria specified in the query filter. The **WSChildSubnetBlock** structure consists of two substructures (**WSChildBlock** and **WSSubnetPolicy**) which can then be modified and/or imported using the **importChildBlock** API. The format of the **WSChildBlock** and the **WSSubnetPolicy** match that defined by the **importChildBlock**.

WSChildSubnetBlock

Below is the portion of *Exports.wsdl* that describes **WSChildSubnetBlock**.

```
<complexType name="WSChildSubnetBlock">
  <sequence>
    <element name="childBlock" nillable="true" type="tns1:WSChildBlock"/>
    <element name="subnetPolicy" nillable="true" type="tns1:WSSubnetPolicy"/>
  </sequence>
</complexType>
```

Element	Description and accepted values
childBlock	The WSChildBlock substructure (see below)
subnetPolicy	The WSSubnetPolicy substructure (see below)

WSChildBlock

Below is the portion of *Exports.wsdl* that describes **WSChildBlock**, the first parameter structure passed to **importChildBlock**. The elements are described in the table that follows.

```
<complexType name="WSChildBlock">
  <sequence>
    <element name="SWIPname" nillable="true" type="soapenc:string" />
    <element name="allocationReason" nillable="true" type="soapenc:string" />
    <element name="allocationReasonDescription" nillable="true" type="soapenc:string" />
    <element name="allocationTemplate" nillable="true" type="soapenc:string" />
    <element name="blockAddr" nillable="true" type="soapenc:string" />
    <element name="blockName" nillable="true" type="soapenc:string" />
    <element name="blockSize" nillable="true" type="soapenc:string" />
    <element name="blockStatus" nillable="true" type="soapenc:string" />
    <element name="blockType" nillable="true" type="soapenc:string" />
    <element name="container" nillable="true" type="soapenc:string" />
    <element name="createReverseDomains" nillable="true" type="soapenc:string" />
    <element name="description" nillable="true" type="soapenc:string" />
    <element name="domainType" nillable="true" type="soapenc:string" />
    <element name="excludeFromDiscovery" nillable="true" type="soapenc:string"/>
    <element name="interfaceAddress" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="interfaceName" nillable="true" type="soapenc:string" />
    <element name="ipv6" type="xsd:boolean"/>
    <element name="primarySubnet" type="xsd:boolean"/>
    <element name="nonBroadcast" type="xsd:boolean"/>
    <element name="userDefinedFields" nillable="true" type="impl:ArrayOf_soapenc_string" />
```

```
  </sequence>
</complexType>
```

Element	Description and accepted values
SWIPname	SWIP name for this block
allocationReason	The name of a pre-existing Allocation Reason.
allocationReasonDescription	A description of the reason for the allocation.
allocationTemplate	Reserved
blockAddr	The starting address of the block.
blockName	The name of the block.
blockSize	The size of the block in short-notation (e.g., 24 for a 255.255.255.0 network).
blockStatus	The current status of the block. Possible values are: Deployed, FullyAssigned, Reserved, Aggregate.
blockType	The Block Type for the block.
container	The name of the container holding the block.
createReverseDomains	Whether or not to automatically create reverse DNS domain(s) for this block. Accepted values are true or false. If not specified, defaults to false.
description	The description of the block.
domainType	The domain type of the reverse DNS domain.
excludeFromDiscovery	True when this subnet excluded from Host Discovery tasks.
interfaceAddress	The address(es) of the interface IP address.
interfaceName	If this block is in a device container, the name of the interface to which it's attached.
ipv6	True if this is an IPV6 block.
primarySubnet	True if this is a primary subnet.
userDefinedFields	A string array containing one or more name=value pairs, where the name is the UDF name and the value is the desired value, for example, State=PA.

WSSubnetPolicy

The portion of *Exports.wsdl* that describes WSSubnetPolicy, the second parameter structure passed to **`ImportChildBlock`** is shown below. The elements are described in the table that follows.

```
<complexType name="WSSubnetPolicy">
  <sequence>
    <element name="DHCPOptionsSet" nillable="true" type="soapenc:string" />
    <element name="DHCPPolicySet" nillable="true" type="soapenc:string" />
    <element name="DNSServers" nillable="true" type="impl:ArrayOf_soapenc_string" />
    <element name="cascadePrimaryDhcpServer" nillable="true" type="xsd:boolean" />
    <element name="defaultGateway" nillable="true" type="soapenc:string" />
    <element name="failoverDHCPServer" nillable="true" type="soapenc:string" />
    <element name="forwardDomainTypes" nillable="true" type="impl:ArrayOf soapenc string"/>
    <element name="forwardDomains" nillable="true" type="impl:ArrayOf_soapenc_string" />
    <element name="networkLink" nillable="true" type="soapenc:string"/>
    <element name="primaryDHCPServer" nillable="true" type="soapenc:string" />
    <element name="primaryWINSServer" nillable="true" type="soapenc:string" />
    <element name="reverseDomainTypes" nillable="true" type="impl:ArrayOf_soapenc_string"/>
```

```
    <element name="reverseDomains" nillable="true" type="impl:ArrayOf_soapenc_string" />
  </sequence>
</complexType>
```

Element	Description and accepted values
cascadePrimaryDHCPServer	Import only.
DHCPOptionsSet	The name of a previously defined DHCP Options set.
DHCPPolicySet	The name of a previously defined DHCP policy set.
DNSServers	The name of previously defined DNS servers to be sent as an IP address to the client.
defaultGateway	The default gateway that DHCP clients on this subnet will use. Accepted value is an IP address.
failoverDHCPServer	The name of the DHCP server that will act as failover for this subnet. This cannot be the same as the primary DHCP server.
forwardDomains	The forward domain names that will available to the user when adding an IP address to the system. The first forward domain in the list will be used when there is a domain name DHCP option.
forwardDomainTypes	The domain types corresponding to the domains listed in forwardDomains. Only required for non-default domain types.
networkLink	The name of the Shared Network Segment for this subnet.
primaryDHCPServer	The name of the DHCP server that will act as primary for this subnet.
primaryWINSServer	The IP address of the Microsoft WINS server for clients in this subnet.
reverseDomains	The reverse domain names that will available to the user when adding an IP address to the system.
reverseDomainTypes	The domain types corresponding to the domains listed in reverseDomains. Only required for non-default domain types.

Optional Service

After the **`initExportChildBlock`** API is called to initialize the API, the web service client can*, optionally,* call the **`initExportChildBlockUDFTags`** API. This service is used by the ExportChildBlock CLI to create the header line used when exporting with the expanded format option. The portion of *Exports.wsdl* that describes the **`initExportChildBlockUDFTags`** request and response messages is shown below.

```
<wsdl:message name="initExportChildBlockUDFTagsRequest">
   <wsdl:part name="context" type="tns2:WSContext"/>
</wsdl:message>
<wsdl:message name="initExportChildBlockUDFTagsResponse">
  <wsdl:part name="initExportChildBlockUDFTagsReturn" type="impl:ArrayOf_soapenc_string"/>
</wsdl:message>
```

Request

The **`WSContext`** passed as input by the client web service is the **`WSContext`** object returned by the **`initExportChildBlock`** service defined above.

Response

The result returned from the **`initExportChildBlockUDFTags`** service is an array of strings. These are the field names/tags of the user defined fields defined for the blocks that will be returned on subsequent calls to the **`exportChildBlock`** service.

ExportContainer

Overview

The **exportContainer** API enables the web service client to issue a request to retrieve a list of Containers from IPAM. This service enables the client to filter the list of Containers retrieved.

Initialization

Before the **exportContainer** API is called, the web service client *must* call **initExportContainer** to initialize the API. The portion of *Exports.wsdl* that describes the **initExportContainer** request and response messages is shown below.

```
<wsdl:message name="initExportContainerRequest">
  <wsdl:part name="query" type="soapenc:string"/>
  <wsdl:part name="options" type="impl:ArrayOf_soapenc_string"/>
</wsdl:message>
<wsdl:message name="initExportContainerResponse">
  <wsdl:part name="initExportContainerReturn" type="tns2:WSContext"/>
</wsdl:message>
```

Request

The query string is passed as input from the client to the **initExportContainer** web service. The query string syntax is defined previously. Supported selectors for exporting devices are defined in the following table.

Selector	Description	Example
Name	Exports container by container name.	Name='west; Name ends 'est' Name begins 'wes' Name contains 'es'
User Defined Fields	Exports container by user defined field name and value. Usage UDF.<fieldname>='<fieldvalue>'	UDF.order='first' UDF.order begins 'fir' UDF.order ends 'rst' UDF.order contains 'irs'

The options array is used to pass additional option information to the service. The valid options for **ExportContainer** are described in the following table:

Option	Description and accepted values
ParentContainerFullPath	When this option is specified, the service populates the parent container field using the long format, for example: InControl/Texas/Dallas

Response

The response from the **initExportContainer** web service is a **WSContext** object defined previously and *must* be included in each successive call to **exportContainer**, as described below.

Service Invocation

The portion of *Exports.wsdl* that describes the **exportContainer** request and response messages is shown below.

```
<wsdl:message name="exportContainerRequest">
  <wsdl:part name="context" type="tns1:WSContext"/>
</wsdl:message>
<wsdl:message name="exportContainerResponse">
  <wsdl:part name="exportContainerReturn" type="impl:ArrayOf_tns2_WSContainer"/>
</wsdl:message>
```

Request

The **WSContext** passed as input by the client web service is the WSContext object returned by the **initExportContainer** service defined above. This **WSContext** has the **maxResults** field set to a default value of 100. When this context is provided to a subsequent call to **exportContainer**, the number of exported containers is limited to the first 100 that match the criteria in the given query filter. The web service client may change this **maxResults** attribute of the **WSContext** before any call to the **exportContainer** service to modify the size of the resultant **WSContainer** object array. However, the value specified by the client cannot exceed 100.

Response

The result returned from the **exportContainer** service is an array of **WSContainer** objects matching the selection criteria specified in the query filter. The **WSContainer** can then be modified and/or imported using the **importContainer** API. The **WSContainer** object is described in the **ImportContainer** API section on page 272.

ExportDevice

Overview

The **exportDevice** API enables the web service client to issue a request to retrieve a list of Devices from IPAM. This service enables the client to filter the list of Devices retrieved.

Initialization

Before the **exportDevice** API is called, the web service client *must* call **initExportDevice** to initialize the API. The portion of *Exports.wsdl* that describes the **initExportDevice** request and response messages is shown following.

```
<wsdl:message name="initExportDeviceRequest">
  <wsdl:part name="filter" type="soapenc:string"/>
  <wsdl:part name="options" type="impl:ArrayOf_soapenc_string"/>
</wsdl:message>
<wsdl:message name="initExportDeviceResponse">
  <wsdl:part name="initExportDeviceReturn" type="tns2:WSContext"/>
</wsdl:message>
```

Request

The query string is passed as input from the client to the **initExportDevice** web service. The query string syntax is defined previously. Supported selectors for exporting devices are defined in the following table.

In addition, the **initExportDevice** service accepts the options array, described following the selectors table.

Selector	Description	Example
Name	Exports device by hostname.	Name='host; Name ends 'ost' Name begins 'hos' Name contains 'os'
Container	Exports device by Container name.	Container='Exton'; Container ends 'ton' Container begins 'Ext Container contains 'xto'
IP Address	Exports device by IP Address. ***Note***, this filter should not be used on a multi-homed device, unless the desired result is to export the device with **only** the IP Address specified in the Selector filter. Instead, use IPAddressRange, Name, or multiple IPAddress (separated by Or) Selector type filters.	IPAddress=10.0.0.1
IP Address Range	Exports device by IP Address range.	IPAddressRange=10.0.0.1-11.0.0.1

Selector	Description	Example
Device type	Exports device by Device type.	DeviceType='Router' DeviceType ends 'uter' DeviceType begins 'Rou' DeviceType contains 'out'
Domain	Exports device by Domain name.	Domain='ins.com.' Domain begins 'ins.c' Domain ends '.com.' Domain contains 's.com'
Block	Exports device by Block name.	Block='10.0.0.0/24' Block begins '10.0.0' Block ends '.0/24' Block contains '.0.0.0'
Block Type	Exports device by Block type.	BlockType='Any' BlockType begins 'An' BlockType ends 'ny' BlockType contains 'n'
User Defined Fields	Exports device by user defined field name and value. Usage UDF.<fieldname>='<fieldvalue>'	UDF.order='first' UDF.order begins 'fir' UDF.order ends 'rst' UDF.order contains 'irs'
Address Type	Exports device by address type. Specify the numeric value as follows: 1 : Dynamic DHCP 2 : Automatic DHCP 3 : Manual DHCP 4 : Static 5 : Reserved 6 : Dynamic NA DHCPv6 7 : Automatic NA DHCPv6 8 : Manual NA DHCPv6 9 : Dynamic TA DHCPv6 10 : Automatic TA DHCPv6 11 : Manual TA DHCPv6 12 : Dyanmic PD DHCPv6 13 : Automatic PD DHCPv6 14 : Interface	addrType=4 (Static)
Virtual	Exports device by virtual flag. When true is indicated, devices will be exported when an associated IP address is virtual.	Virtual=1 (for true) Virtual=0 (for false)
Circuit ID	Exports device by relay agent circuit ID. The circuit ID needs to be specified in hex encoded format.	relayagentcircuitid='636972637569742d686f73742d61632d31332d31302d32622d30302d3033' relayagentcircuitid ends '033' relayagentcircuitid begins ' 636' relayagentcircuitid contains ' 1332'

Selector	Description	Example
Remote ID	Exports device by relay agent remote ID. The remote ID needs to be specified in hex encoded format.	relayagentremoteid=' 5313102b0003' relayagentremoteid ends '003' relayagentremoteid begins ' 53' relayagentremoteid contains ' 102b'

The options array is used to pass additional option information to the service. The valid options for **ExportDevice** are described in the following table:

Option	Description and accepted values
recurseContainerHierarchy	When this option is specified, the service recursively exports all devices within all child containers specified within the Container Selector filter. This flag is ignored if a Container Selector is not included.

Response

The response from the **initExportDevice** web service is a **WSContext** object defined previously and *must* be included in each successive call to **exportDevice**, as described below.

Service Invocation

The portion of *Exports.wsdl* that describes the **exportDevice** request and response messages is shown following.

```
<wsdl:message name="exportDeviceRequest">
  <wsdl:part name="context" type="tns2:WSContext" />
</wsdl:message>
<wsdl:message name="exportDeviceResponse">
  <wsdl:part name="exportDeviceReturn" type="impl:ArrayOf_tns2_WSDevice"/>
</wsdl:message>
```

Request

The **WSContext** passed as input by the client web service is the **WSContext** object returned by the **initExportDevice** service defined aboveand has the **maxResults** field set to a default value of 100. When this context is provided to a subsequent call to **exportDevice**, the number of exported blocks is limited to the first 100 *or less* (see Paging), that match the criteria in the given query filter. The web service client may change this **maxResults** attribute of the **WSContext** before any call to the **exportDevice** service to modify the size of the resultant **WSDevice** object array. However, the value specified by the client cannot exceed 100.

Paging

A device in IPAM is normalized within the database and thus may be represented by more than a single row in multiple tables. Because of this and for performance, the **exportDevice** cannot guarantee that the number of **WSDevice** objects returned in any single execution of the service will be equal to the max results set on the **WSContext** object. It will, however, always guarantee the number of results to be the max results value or less.

Response

The result returned from the **exportDevice** service is an array of **WSDevice** objects matching the selection criteria specified in the query filter. The **WSDevices** can then be modified and/or imported using the **importDevice** API. The format of the **WSDevice** matches that defined by the **importDevice**.

WSDevice

Below is the portion of *Exports.wsdl* that describes **WSDevice**, the array of structures returned by **exportDevice**. The elements are described in the table that follows.

```
<complexType name="WSDevice">
  <sequence>
    <element name="MACAddress" nillable="true" type="soapenc:string"/>
    <element name="addressType" nillable="true" type="soapenc:string"/>
    <element name="aliases" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="container" nillable="true" type="soapenc:string"/>
    <element name="description" nillable="true" type="soapenc:string"/>
    <element name="deviceType" nillable="true" type="soapenc:string"/>
    <element name="domainName" nillable="true" type="soapenc:string"/>
    <element name="domainType" nillable="true" type="soapenc:string"/>
    <element name="duid" nillable="true" type="soapenc:string"/>
    <element name="dupWarning" nillable="true" type="soapenc:string"/>
    <element name="hostname" nillable="true" type="soapenc:string"/>
    <element name="hwType" nillable="true" type="soapenc:string"/>
    <element name="ipAddress" nillable="true" type="soapenc:string"/>
    <element name="resourceRecordFlag" nillable="true" type="soapenc:string"/>
    <element name="userDefinedFields" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element maxOccurs="unbounded" name="interfaces" nillable="true" type="tns2:WSInterface"/>
    <element name="id" nillable="true" type="soapenc:int"/>
    <element name="excludeFromDiscovery" nillable="true" type="soapenc:string"/
  </sequence>
</complexType>

<complexType name="WSInterface">
  <sequence>
    <element name="addressType" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="container" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="excludeFromDiscovery" nillable="true" type="soapenc:string"/>
    <element name="hwType" nillable="true" type="soapenc:string"/>
    <element name="id" nillable="true" type="soapenc:int"/>
    <element name="ipAddress" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="macAddress" nillable="true" type="soapenc:string"/>
    <element name="name" nillable="true" type="soapenc:string"/>
    <element name="relayAgentCircuitId" nillable="true"
type="impl:ArrayOf_soapenc_string"/>
     <element name="relayAgentRemoteId" nillable="true"
type="impl:ArrayOf_soapenc_string"/>
    <element name="sequence" nillable="true" type="soapenc:int"/>
    <element name="virtual" nillable="true" type="impl:ArrayOf_soapenc_boolean"/>
  </sequence>
</complexType>
```

Element	Accepted Values
MACAddress	The hardware MAC address of the device. DEPRECATED: This is now returned in the WSInterface structure for all devices.
addressType	The address type of this device. Accepted values are: **Static**, **Dynamic DHCP**, **Automatic DHCP**, **Manual DHCP**, **Reserved, Dynamic NA DHCPv6, Automatic NA DHCPv6, Dynamic TA DHCPv6, and**

Element	Accepted Values
	Automatic TA DHCPv6
Aliases	A string array containing the alias or list of aliases for this hostname. When you specify an alias, a CNAME record is created. The alias may be fully qualified (contains a trailing dot), or not. When fully qualified, everything that is after the first qualifier is interpreted as a domain name. When not fully qualified, the CNAME record will be created in the same domain as the device. To use this element, you must also specify resourceRecordFlag as **true**.
container	The name of the container that contains the device. DEPRECATED: This is now returned in the WSInterface structure for all devices.
description	A description of the device.
deviceType	The name of a device type configured in IPAM.
domainName	Domain name already defined to IPAM
domainType	Domain type already defined to IPAM. If not specified, the "Default" domain type will be used.
duid	DHCP Unique Identifier
dupWarning	If the administrator policy of the user indicates "Warn" for the "Allow Duplicate Hostnames Checking" option, the warning will be ignored and the device added with the duplicate hostname when this field is **true**. Accepted values are **true** or **false**. If not specified, defaults to **false**.
hostname	Valid host name or **APPLYNAMINGPOLICY**.
hwtype	Specify **Ethernet** or **Token Ring**. When hwtype is specified, MACAddress must also be specified. DEPRECATED: This is now returned in the WSInterface structure for all devices.
ipAddress	The IP Address of the device. DEPRECATED: This is now returned in the WSInterface structure for all devices.
resourceRecordFlag	Whether or not to add resource records for this device. If not specified as **true**, defaults to false.
userDefinedFields	A string array containing one or more *name=value* pairs, where the *name* is the UDF name and the *value* is the desired value, for example**, State=PA**. If the UDF type is Checkbox, the valid values are **on** and **off**.
Interfaces	An array of WSInterface structures. Each element in the array corresponds to one interface for a multi-homed device, or the single interface for a single-homed device. The fields in the WSInterface structure are listed below. Except where noted, their value is the same as in the WSDevice structure. - addressType - container - excludeFromDiscovery - hwType - id: The internal ID; provide this on a modify request - ipAddress - macAddress - name: Interface Name - relayAgentCircuitId - relayAgentRemoteId - sequence: reserved - virtual: boolean indicating whether this interface address is virtual (See

Element	Accepted Values
	Guide to Using IPAM, "Configuring Shared and Virtual IP Addresses" for more information.)
Id	Internal id; provide this on a modify request
excludeFromDiscovery	Flag indicating if this device should be included in Host Discovery tasks. Accepted values are **true** or **false**. If not specified, defaults to **false**. DEPRECATED: This is now returned in the WSInterface structure for all devices.

Optional Service

After the **`initExportDevice`** API is called to initialize the API, the web service client can, *optionally,* call the **`initExportDeviceUDFTags`** API. This service is used by the ExportDevice CLI to create the header line used when exporting with the expanded format option. The portion of *Exports.wsdl* that describes the **`initExportDeviceUDFTags`** request and response messages is shown below.

```
<wsdl:message name="initExportDeviceUDFTagsRequest">
   <wsdl:part name="context" type="tns2:WSContext"/>
</wsdl:message>
<wsdl:message name="initExportDeviceUDFTagsResponse">
   <wsdl:part name="initExportDeviceUDFTagsReturn" type="impl:ArrayOf_soapenc_string"/>
</wsdl:message>
```

Request

The **`WSContext`** passed as input by the client web service is the **`WSContext`** object returned by the **`initExportDevice`** service defined above.

Response

The result returned from the **`initExportDeviceUDFTags`** service is an array of strings. These are the field names/tags of the user defined fields defined for the devices that will be returned on subsequent calls to the **`exportDevice`** service.

ExportDeviceResourceRecord

Overview

The **exportDeviceResourceRecord** API enables the web service client to issue a request to retrieve a list of resource records for a device or list of devices from IPAM. This service enables the client to filter the list of resource records retrieved by device.

Initialization

Before the **exportDeviceResourceRecord** API is called, the web service client *must* call **initExportDeviceResourceRec** to initialize the API. The portion of *Exports.wsdl* that describes the **initExportDeviceResourceRec** request and response messages is shown following.

```
<wsdl:message name="initExportDeviceResourceRecRequest">
  <wsdl:part name="filter" type="soapenc:string"/>
  <wsdl:part name="options" type="impl:ArrayOf_soapenc_string"/>
</wsdl:message>
<wsdl:message name="initExportDeviceResourceRecResponse">
  <wsdl:part name="initExportDeviceResourceRecReturn" type="tns2:WSContext"/>
</wsdl:message>
```

Request

The query string is passed as input from the client to the **initExportDeviceResourceRec** web service in the filter parameter. The query string syntax is defined previously. Supported selectors for exporting device resource records by device are defined in the following table.

In addition, the **initExportDeviceResourceRec** service accepts an option that specifies that recursively all devices within all child containers specified within the Container Selector filter should be selected. Specify the option parameter as **recurseContainerHierarchy**.

Selector	Description	Example
Name	Select device by hostname.	Name='host; Name ends 'ost' Name begins 'hos' Name contains 'os'
Container	Select device by Container name.	Container='Exton'; Container ends 'ton' Container begins 'Ext Container contains 'xto'
IP Address	Select device by IP Address. ***Note***, this filter should not be used on a multi-homed device, unless the desired result is to export the device with **only** the IP Address specified in the Selector filter. Instead, use IPAddressRange, Name, or multiple IPAddress (separated by Or) Selector type filters.	IPAddress=10.0.0.1
IP Address Range	Select device by IP Address range.	IPAddressRange=10.0.0.1-11.0.0.1

Selector	Description	Example
Device type	Select device by Device type.	DeviceType='Router' DeviceType ends 'uter' DeviceType begins 'Rou' DeviceType contains 'out'
Domain	Select device by Domain name.	Domain='ins.com.' Domain begins 'ins.c' Domain ends '.com.' Domain contains 's.com'
Domain Type	Select device by domain type.	DomainType='Internal' DomainType begins 'abc' DomainType ends 'xyz' Domain Type contains 'lmnop'
Block	Select device by Block name.	Block='10.0.0.0/24' Block begins '10.0.0' Block ends '.0/24' Block contains '.0.0.0'
Block Type	Select device by Block type.	BlockType='Any' BlockType begins 'An' BlockType ends 'ny' BlockType contains 'n'
User Defined Fields	Select device by user defined field name and value. Usage UDF.<fieldname>='<fieldvalue>'	UDF.order='first' UDF.order begins 'fir' UDF.order ends 'rst' UDF.order contains 'irs'

Response

The response from the **`initExportDeviceResourceRec`** web service is a **`WSContext`** object defined previously and *must* be included in each successive call to **`exportDeviceResourceRec`**, as described below.

Service Invocation

The portion of *Exports.wsdl* that describes the **`exportDeviceResourceRec`** request and response messages is shown following.

```
<wsdl:message name="exportDeviceResourceRecRequest">
  <wsdl:part name="context" type="tns2:WSContext" />
</wsdl:message>
<wsdl:message name="exportDeviceResourceRecResponse">
  <wsdl:part name="exportDeviceResourceRecReturn" type="impl:ArrayOf_tns2_WSDeviceResourceRec"/>
</wsdl:message>
```

Request

The **WSContext** passed as input by the client web service is the **WSContext** object returned by the **initExportDeviceRec** service defined above and has the **maxResults** field set to a default value of 5000. When this context is provided to a subsequent call to **exportDeviceResourceRec**, the number of exported resource records is limited to the first 5000 devices, *or less* (see Paging), that match the criteria in the given query filter. The web service client may change this **maxResults** attribute of the **WSContext** before any call to the **exportDeviceResourceRec** service to modify the size of the resultant **WSDeviceResourceRec** object array. However, the value specified by the client cannot exceed 5000.

Paging

A device in IPAM is normalized within the database and thus may be represented by more than a single row in multiple tables. Because of this and for performance, the **exportDeviceResourceRec** cannot guarantee that the number of **WSDeviceResourceRec** objects returned in any single execution of the service will be equal to the max results set on the **WSContext** object. It will, however, always guarantee the number of results to be the max results value or less.

Response

The result returned from the **exportDeviceResourceRec** service is an array of **WSDeviceResourceRec** objects matching the selection criteria specified in the query filter. The **WSDeviceResourceRecs** can then be modified and/or imported using the **importDeviceResourceRecord** API. The format of the **WSDeviceResourceRec** matches that defined by the **importDeviceResourceRecord**.

WSDeviceResourceRec

The portion of *Exports.wsdl* that describes **WSDeviceResourceRec**, the array of structures returned by **exportDeviceResourceRec** is shown following. The elements are described in the table that follows.

```
<complexType name="WSDeviceResourceRec">
  <sequence>
    <element name="TTL" nillable="true" type="soapenc:string"/>
    <element name="comment" nillable="true" type="soapenc:string"/>
    <element name="container" nillable="true" type="soapenc:string"/>
    <element name="data" nillable="true" type="soapenc:string"/>
    <element name="domain" nillable="true" type="soapenc:string"/>
    <element name="domainType" nillable="true" type="soapenc:string"/>
    <element name="hostname" nillable="true" type="soapenc:string"/>
    <element name="id" nillable="true" type="soapenc:int"/>
    <element name="ipAddress" nillable="true" type="soapenc:string"/>
    <element name="owner" nillable="true" type="soapenc:string"/>
    <element name="resourceRecClass" nillable="true" type="soapenc:string"/>
    <element name="resourceRecType" nillable="true" type="soapenc:string"/>
  </sequence>
</complexType>
```

Element	Accepted Values
container	The name of the container that holds the device. This is required only if there is overlapping address space in use and the ip address is in overlapping space. The container is then used to uniquely determine the device.
domainType	Domain type already defined to IPAM. If not specified, the "Default" domain type will be used.
domain	Domain name where resource records are to be added.
hostname	The device host name.
ipAddress	The IP Address of the Device.
owner	The owner field of the resource record.
resourceRecClass	The Class of the Resource Record. Defaults to "IN".
resourceRecordType	The Type of the resource Record.
TTL	The Time To Live for the record.
data	The data portion of the resource record. The format is dependent on the type specified above.
comment	Comment text associated with the resource record.

ExportDeviceRestoreList

Overview

The **exportDeviceRestoreList** API enables the web service client to issue a request to retrieve a list of devices that have been deleted and may be eligible for restoring. This service enables the client to filter the list of devices(deleted) exported. A superuser may filter the results based on the requesting administrator(not available for non-superusers). All users may filter the list based on IP Address, hostname, block name, container, address type, device type, or IP address range

Initialization

Before the **exportDeviceRestoreList** API is called, the web service client *must* call **initExportDeviceRestoreList** to initialize the API. The portion of *Exports.wsdl* that describes the **initExportDeviceRestoreList** request and response messages is shown following.

```
<wsdl:message name="initExportDeviceRestoreListRequest">
      <wsdl:part name="filter" type="soapenc:string"/>
      <wsdl:part name="options" type="impl:ArrayOf_soapenc_string"/>
</wsdl:message>
<wsdl:message name="initExportDeviceRestoreListResponse">
      <wsdl:part name="initExportDeviceRestoreListReturn" type="tns2:WSContext"/>
</wsdl:message>
```

Request

The query string is passed as input from the client to the **initExportDeviceRecordList** web service in the filter parameter. The query string syntax is defined previously. Supported selectors for exporting device resource records by device are defined in the following table.

Currently, there are no options defined for this service.

Selector	Description	Example
Admin	Export devices deleted by a particular user. This filter is only available for superusers.	Admin='someuser' Admin ends 'ser' Admin begins 'some' Admin contains 'ome'
Name	Export deleted devices with a particular hostname.	Name='host; Name ends 'ost' Name begins 'hos' Name contains 'os'
Ipaddress	Export deleted devices with a particular IP address.	Ipaddress='168.0.0.1'
Block	Export devices deleted from a particular block.	Block='10.0.0.0/24' Block begins '10.0.0'

Selector	Description	Example
		Block ends '.0/24' Block contains '.0.0.0'
Container	Export devices deleted from a particular container.	Container='Exton' Container='/InControl/USA/Exton' Container ends 'ton' Container begins 'Ext Container contains 'xto'
DeviceType	Export deleted devices of a particular device type.	DeviceType='Router' DeviceType ends 'uter' DeviceType begins 'Rou' DeviceType contains 'out'
HWAddress	Export deleted devices of a particular Hardware address	HWAddress='000000000ABC' HWAddress ends 'ABC' HWAddress begins '0000000' HWAddress contains '00A'
IPAddressRange	Export deleted devices with IP addresses that fall in a particular range.	IPAddressRange=10.0.0.1-10.0.0.100
addrType	Exports device by address type. Specify the numeric value as follows: 1 : Dynamic DHCP 2 : Automatic DHCP 3 : Manual DHCP 4 : Static 5 : Reserved 6 : Dynamic NA DHCPv6 7 : Automatic NA DHCPv6 8 : Manual NA DHCPv6 9 : Dynamic TA DHCPv6 10 : Automatic TA DHCPv6 11 : Manual TA DHCPv6 12 : Dyanmic PD DHCPv6 13 : Automatic PD DHCPv6 14 : Interface	addrType=4 (Static)

Response

The response from the **`initExportDeviceRestoreList`** web service is a **`WSContext`** object defined previously. This **`WSContext`** object *must* be included in each successive call to **`exportDeviceRestoreList`** , as described below.

Service Invocation

The portion of *Exports.wsdl* that describes the **`exportDeviceRestoreList`** request and response messages is shown following.

```
<wsdl:message name="exportDeviceRestoreListRequest">
   <wsdl:part name="context" type="tns2:WSContext"/>
</wsdl:message>
<wsdl:message name="exportDeviceRestoreListResponse">
```

```
   <wsdl:part name="exportDeviceRestoreListReturn"
   type="impl:ArrayOf_tns2_WSRestoreDevice"/>
</wsdl:message>
```

Request

The **WSContext** passed as input by the client web service is the **WSContext** object returned by the **initExportDeviceRestoreList** service defined above and has the **maxResults** field set to a default value of 5000. When this context is provided to a subsequent call to **exportDeviceRestoreList**, the number of exported resource records is limited to the first 5000 resource record change requests, or less (see Paging), that match the criteria in the given query filter. The web service client may change this **maxResults** attribute of the **WSContext** before any call to the **exportDeviceRestoreList** service to modify the size of the resultant **WSRestoreDevice** object array. However, the value specified by the client cannot exceed 5000.

Paging

A resource record change request in IPAM is normalized within the database and thus may be represented by more than a single row in multiple tables. Because of this and for performance, the **exportDeviceRestoreList** service cannot guarantee that the number of **WSDeviceRecord** objects returned in any single execution of the service will be equal to the max results set on the **WSContext** object. It will, however, always guarantee the number of results to be the max results value or less.

Response

The result returned from the **exportDeviceRestoreList** service is an array of **WSRestoreDevice** objects matching the selection criteria specified in the query filter. **WSRestoreDevice** thus returned can then be used to invoke the **restoreDeletedDevice** API.

WSRestoreDevice

The portion of *Exports.wsdl* that describes **WSRestoreDevice**, the array of structures returned by **exportDeviceRestoreList** is shown following. The elements are described in the table that follows.

```
<complexType name="WSRestoreDevice">
    <sequence>
     <element name="MACAddress" nillable="true" type="soapenc:string"/>
     <element name="addressType" nillable="true" type="soapenc:string"/>
     <element name="adminLoginId" nillable="true" type="soapenc:string"/>
     <element name="container" nillable="true" type="soapenc:string"/>
     <element name="description" nillable="true" type="soapenc:string"/>
     <element name="deviceType" nillable="true" type="soapenc:string"/>
     <element name="domainName" nillable="true" type="soapenc:string"/>
     <element name="domainType" nillable="true" type="soapenc:string"/>
     <element name="duid" nillable="true" type="soapenc:string"/>
     <element name="hostname" nillable="true" type="soapenc:string"/>
     <element name="hwType" nillable="true" type="soapenc:string"/>
     <element name="ignoreDuplicateWarning" nillable="true" type="soapenc:string"/>
     <element name="ipAddress" nillable="true" type="soapenc:string"/>
     <element name="restoreId" nillable="true" type="soapenc:int"/>
     <element name="blockName" nillable="true" type="soapenc:string"/>
     <element name="dateTime" nillable="true" type="soapenc:string"/>
    </sequence>
```

```
</complexType>
```

Element	Accepted Values
MACAddress	The Hardware(Mac)Address of the deleted device.
addressType	Address type of the deleted device.
adminLoginId	The login id of the administrator who deleted the record.
container	Name of the Container from which the device was deleted.
description	Description of the deleted device.
deviceType	Device type of the deleted device
domainName	Domain name of the deleted device.
domainType	Domain type of the deleted device.
Duid	DHCP Unique Identifier
Hostname	The hostname of the deleted device.
hwType	
ignoreDuplicateWarning	This field is used by RestoreDeletedDevice API. Export puts a false by default for this. Change this to True before using this output for restoring to ignore Duplicate Warnings for mac address or hostname.
ipAddress	The ipaddress of the deleted device.
restoreId	The id used internally to identify the record. It is required to restore this record via RestoreDeletedDevice.
blockName	The name of the block that the device was deleted from.
dateTime	The date and time when the device was deleted.

ExportDomainResourceRecord

Overview

The **exportDomainResourceRecord** API enables the web service client to issue a request to retrieve a list of resource records for a domain or list of domains from IPAM. This service enables the client to filter the list of resource records retrieved by domain and other attributes of the resource record.

The default behavior of this API is to export only those resource records matching the selectors and not bound to a device . This information can then be used with the **ImportDomainResourceRecord** API, which allows the administrator to create and modify resource records that are not bound to a particular device.

When the "-id" option (Include Device RRs) is specified , this export will include all of the resource records that are visible on the domain's Resource Record tab in the IPAM user interface, including those bound to a device. Be aware that if you re-import resource records bound to a device using the **ImportDomainResourceRecord** API, the association with the device will be lost.

Initialization

Before the **exportDomainResourceRecord** API is called, the web service client *must* call **initExportDomainResourceRec** to initialize the API. The portion of *Exports.wsdl* that describes the **initExportDomainResourceRec** request and response messages is shown following.

```
<wsdl:message name="initExportDomainResourceRecRequest">
  <wsdl:part name="filter" type="soapenc:string"/>
  <wsdl:part name="options" type="impl:ArrayOf_soapenc_string"/>
</wsdl:message>
<wsdl:message name="initExportDomainResourceRecResponse">
  <wsdl:part name="initExportDomainResourceRecReturn" type="tns2:WSContext"/>
</wsdl:message>
```

Request

The query string is passed as input from the client to the **initExportDomainResourceRec** web service in the filter parameter. The query string syntax is defined previously. Supported selectors for exporting domain resource records by domain are defined in the following table. All of the selectors support "=", "begins", "ends" and "contains".

Selector	Description	Example
Domain	Select resource records by Domain name.	Domain='ins.com.' Domain begins 'ins.c' Domain ends '.com.' Domain contains 's.com'
Domain Type	Select resource records by Domain type.	DomainType='Internal'

Selector	Description	Example
User Defined Fields	Select domain for resource records by user defined field name and value. Usage UDF.<fieldname>='<fieldvalue>'	UDF.order contains 'irs'
Owner	Select resource records by owner.	"owner begins 's'"
RR type	Select resource records by record type.	RR_type='CNAME'
RR data	Select resource records by the record data contents.	rdata='10.0.0.50'

The options array is used to pass additional option information to the service. The valid options for **`ExportDomainResourceRec`** are described in the following table:

Option	Description and accepted values
includeDeviceRRs	When this option is specified, all resource records for the included domains will be exported, including those bound to a device. The default behavior is to export only those resource records **not** bound to a device.

Response

The response from the **`initExportDomainResourceRec`** web service is a **`WSContext`** object defined previously and *must* be included in each successive call to **`exportDomainResourceRec`**, as described below.

Service Invocation

The portion of *Exports.wsdl* that describes the **exportDomainResourceRec** request and response messages is shown following.

```
<wsdl:message name="exportDomainResourceRecRequest">
  <wsdl:part name="context" type="tns2:WSContext" />
</wsdl:message>
<wsdl:message name="exportDomainResourceRecResponse">
  <wsdl:part name="exportDomainResourceRecReturn" type="impl:ArrayOf_tns2_WSDomainResourceRec"/>
</wsdl:message>
```

Request

The **WSContext** passed as input by the client web service is the **WSContext** object returned by the **initExportDomainRec** service defined above and has the **maxResults** field set to a default value of 5000. When this context is provided to a subsequent call to **exportDomainResourceRec**, the number of exported resource records is limited to the first 5000 resource records, *or less* (see Paging), that match the criteria in the given query filter. The web service client may change this **maxResults** attribute of the **WSContext** before any call to the **exportDomainResourceRec** service to modify the size of the resultant **WSDomainResourceRec** object array. However, the value specified by the client cannot exceed 5000.

Response

The result returned from the **exportDomainResourceRec** service is an array of **WSDomainResourceRec** objects matching the selection criteria specified in the query filter. The **WSDomainResourceRec** (s) can then be modified and/or imported using the **importDomainResourceRecord** API. The format of the **WSDomainResourceRec** matches that defined by the **importDomainResourceRecord**.

WSDomainResourceRec

The portion of *Exports.wsdl* that describes **WSDomainResourceRec**, the array of structures returned by **exportDomainResourceRec** is shown following. The elements are described in the table that follows.

```
<complexType name="WSDomainResourceRecord">
  <sequence>
    <element name="TTL" nillable="true" type="soapenc:string" />
    <element name="comment" nillable="true" type="soapenc:string" />
    <element name="data" nillable="true" type="soapenc:string" />
    <element name="deviceRecFlag" type="xsd:boolean"/>
    <element name="domain" nillable="true" type="soapenc:string" />
    <element name="domainType" nillable="true" type="soapenc:string" />
    <element name="id" nillable="true" type="soapenc:int"/>
    <element name="owner" nillable="true" type="soapenc:string" />
    <element name="resourceRecClass" nillable="true" type="soapenc:string" />
    <element name="resourceRecType" nillable="true" type="soapenc:string" />
  </sequence>
</complexType>
```

Element	Accepted Values
TTL	The Time To Live for the record.
comment	Comment text associated with the resource record.
data	The data portion of the resource record. The format is dependent on the type specified above.
deviceRecFlag	When **true**, this indicates that the resource record is bound to a device. When **false**, this indicates that the resource record is associated with the domain only, and not a specific device.
domain	Domain name where resource records are associated.
domainType	Domain type of domain in next element.
id	Internal id; provide this on a modify request.
owner	The owner field of the resource record.
resourceRecClass	The Class of the Resource Record.
resourceRecordType	The Type of the resource Record.

ExportNetworkElement

Overview

The **exportNetworkElement** API enables the web service client to issue a request to retrieve a list of Network Elements from IPAM. This service enables the client to filter the list of Network Elements retrieved.

Initialization

Before the **exportNetworkElement** API is called, the web service client *must* call **initExportNetworkElement** to initialize the API. The portion of *Exports.wsdl* that describes the **initExportNetworkElement** request and response messages is shown following.

```
<wsdl:message name="initExportNetworkElementRequest">
      <wsdl:part name="filter" type="soapenc:string"/>
      <wsdl:part name="options" type="impl:ArrayOf_soapenc_string"/>
   </wsdl:message>
 <wsdl:message name="initExportNetworkElementResponse">
      <wsdl:part name="initExportNetworkElementReturn" type="tns2:WSContext"/>
   </wsdl:message>
```

Request

The query string is passed as input from the client to the **initExportNetworkElement** web service. The query string syntax is defined previously. Supported selectors for exporting network elements are defined in the following table.

Selector	Description	Example
Name	Exports network element by Name.	Name='router1.bt.com; Name ends 'com' Name begins 'r' Name contains 'out'
IP Address	Exports network element by IP Address. Also supports ends, begins, contains.	ipaddress=10.0.0.1 ipaddress =2001:db8::1 ipaddress =2001:db8:0:0:0:0:0:1
Vendor	Exports network element by Vendor. Also supports ends, begins, contains.	vendor='Cisco Systems';
Model	Exports network element by Model. Also supports ends, begins, contains.	model=' Cisco 10000 Router';
Type	Exports network element by Type. Also supports ends, begins, contains.	deviceType ='Router'
Agent Name	Exports device by Agent name. Also supports ends, begins, contains.	agent='admin.bt.com'
Global Sync	Exports network elements by global sync setting. Specify 0 for false and 1 for true.	globalsync=0 globalsync=1

Response

The response from the **initExportNetworkElement** web service is a **WSContext** object defined previously, and *must* be included in each successive call to **exportNetworkElement**, as described below.

Service Invocation

Below is the portion of *Exports.wsdl* that describes the **exportNetworkElement** request and response messages.

```
<wsdl:message name="exportNetworkElementRequest">
   <wsdl:part name="in0" type="tns2:WSContext"/>
</wsdl:message>
<wsdl:message name="exportNetworkElementResponse">
   <wsdl:part name="exportNetworkElementReturn" type="impl:ArrayOf_tns2_WSNetworkElement"/>
</wsdl:message>
```

Request

The **WSContext** passed as input by the client web service is the **WSContext** object returned by the **initNetworkElement** service defined above, and has the **maxResults** field set to a default value of 100. When this context is provided to a subsequent call to **exportNetworkElement**, the number of exported network links is limited to the first 100 that match the criteria in the given query filter. The web service client may change this **maxResults** attribute of the **WSContext** before any call to the **exportNetworkElement** service to modify the size of the resultant **WSNetworkElement** object array. However, the value specified by the client cannot exceed 100.

Response

The result returned from the **exportNetworkElement** service is an array of **WSNetworkElement** objects matching the selection criteria specified in the query filter. The **WSNetworkElement** can then be modified and/or imported using the **importNetworkElement** API. The format of the **WSNetworkElement** matches that defined by the **importNetworkElement**.

WSNetworkElement

Below is the portion of *Exports.wsdl* that describes **`WSNetworkElement`**, the array of structures returned by **`exportNetworkElement`**. The elements are described in the table that follows.

```
<complexType name="WSNetworkElement">
    <sequence>
     <element name="SNMPRetries" nillable="true" type="soapenc:int"/>
     <element name="SNMPTimeout" nillable="true" type="soapenc:int"/>
     <element name="SNMPVersion" nillable="true" type="soapenc:string"/>
     <element name="SNMPsysDescr" nillable="true" type="soapenc:string"/>
     <element name="SNMPsysLocation" nillable="true" type="soapenc:string"/>
     <element name="SNMPsysName" nillable="true" type="soapenc:string"/>
     <element name="SNMPsysServices" nillable="true" type="soapenc:string"/>
     <element name="agentNames" nillable="true" type="impl:ArrayOf_soapenc_string"/>
     <element name="description" nillable="true" type="soapenc:string"/>
     <element name="globalSync" type="xsd:boolean"/>
     <element name="id" nillable="true" type="soapenc:int"/>
     <element name="interfaceNames" nillable="true" type="impl:ArrayOf_soapenc_string"/>
     <element name="interfaceStatus" nillable="true" type="impl:ArrayOf_soapenc_string"/>
     <element name="interfaceTemplate" nillable="true" type="soapenc:string"/>
     <element name="ipAddress" nillable="true" type="soapenc:string"/>
     <element name="model" nillable="true" type="soapenc:string"/>
     <element name="name" nillable="true" type="soapenc:string"/>
     <element name="readCommunityString" nillable="true" type="soapenc:string"/>
     <element name="type" nillable="true" type="soapenc:string"/>
     <element name="v3AuthPassword" nillable="true" type="soapenc:string"/>
     <element name="v3AuthProtocol" nillable="true" type="soapenc:string"/>
     <element name="v3ContextName" nillable="true" type="soapenc:string"/>
     <element name="v3EngineId" nillable="true" type="soapenc:string"/>
     <element name="v3PrivacyPassword" nillable="true" type="soapenc:string"/>
     <element name="v3PrivacyProtocol" nillable="true" type="soapenc:string"/>
     <element name="v3Username" nillable="true" type="soapenc:string"/>
     <element name="vendor" nillable="true" type="soapenc:string"/>
    </sequence>
   </complexType>
```

Element	Description and accepted values
SNMPRetries	The number of connection retries that the SNMP Agent will attempt.
SNMPTimeout	The number of milliseconds the SNMP Agent will wait without receiving any messages from its partner before it assumes that the connection to its partner has failed.
SNMPVersion	V1, V2 or V3.
SNMPsysDescr	Populated after a Discover Router Subnets task is run against the network element. This is a standard MIB-II variables (see RFC-1213).
SNMPsysLocation	Populated after a Discover Router Subnets task is run against the network element. This is a standard MIB-II variables (see RFC-1213).
SNMPsysName	Populated after a Discover Router Subnets task is run against the network element. This is a standard MIB-II variables (see RFC-1213).
SNMPsysServices	Populated after a Discover Router Subnets task is run against the network element. This is a standard MIB-II variables (see RFC-1213).

Element	Description and accepted values
agentNames	The name(s) of the IPAM Agent(s) that are responsible for contacting this Network Service. If there are multiple agents, their names are separated by vertical bars("|").
description	A description of the Network Element. "\r\n" is used to separate lines.
globalSync	Whether or not to include this Network Element in the Global Sync process. Value is exported as true or false.
id	Internal id; provide this on a modify request.
interfaceNames	List of interface names.
interfaceStatus	List of each interface status corresponding to the interfaceNames list.
interfaceTemplate	Not exported; only applies to import.
ipAddress	The IP address or fully-qualified domain name (FQDN) of the Network Element.
model	The model name of the Network Element.
name	The name of the Network Element.
readCommunityString	The community string used by SNMP V1, V2 to read details from this network element.
type	The type of Network Element.
v3AuthPassword	If an authorization protocol is being used, exported as "********".
v3AuthProtocol	**MD5**, **SHA1**, or blank.
v3ContextName	SNMP V3 context name.
v3EngineId	SNMP V3 engine id.
v3PrivacyPassword	If a privacy protocol is being used, exported as "********".
v3PrivacyProtocol	**DES** or blank.
v3Username	SNMP V3 user name.
vendor	The vendor of the Network Element.

ExportNetworkLink

Overview

The **exportNetworkLink** API enables the web service client to issue a request to retrieve a list of Network Links from IPAM. This service enables the client to filter the list of Network Links retrieved.

Initialization

Before the **exportNetworkLink** API is called, the web service client *must* call **initExportNetworkLink** to initialize the API. The portion of *Exports.wsdl* that describes the **initExportNetworkLink** request and response messages is shown below.

```
<wsdl:message name="initExportNetworkLinkResponse">
      <wsdl:part name="initExportNetworkLinkReturn" type="tns2:WSContext"/>
</wsdl:message>
<wsdl:message name="initExportNetworkLinkRequest">
      <wsdl:part name="in0" type="soapenc:string"/>
      <wsdl:part name="in1" type="impl:ArrayOf_soapenc_string"/>
</wsdl:message>
```

Request

The query string is passed as input from the client to the **initNetworkLink** web service. The query string syntax is defined previously. Supported selectors for exporting network links are defined in the following table.

Selector	Description	Example
name	The name of the networklink to export. Partial names are supported using the begins/ends/contains qualifiers.	name='My Block' name begins 'My' name ends 'Block' name contains 'Bl'
linktype	Exports device by address type. Specify the numeric value as follows: 0 : physical 1 : logical	linktype=1
description	The description of the networklink to export. Partial descriptions are supported.	description contains 'something'
block	The block name of the block for which the link will be exported.	block begins '10.0'
container	When the block is in overlapping space, use this selector to specify the container. The short name or full path name can be specified.	container='Exton' container=' InControl/Texas/Dallas'

Response

The response from the **initNetworkLink** web service is a **WSContext** object defined previously, and *must* be included in each successive call to **exportNetworkLink**, as described below.

Service Invocation

Below is the portion of *Exports.wsdl* that describes the **exportNetworkLink** request and response messages.

```
<wsdl:message name="exportNetworkLinkRequest">
      <wsdl:part name="in0" type="tns2:WSContext"/>
</wsdl:message>
<wsdl:message name="exportNetworkLinkResponse">
      <wsdl:part name="exportNetworkLinkReturn" type="impl:ArrayOf_tns2_WSNetworkLink"/>
</wsdl:message>
```

Request

The **WSContext** passed as input by the client web service is the **WSContext** object returned by the **initNetworkLink** service defined above, and has the **maxResults** field set to a default value of 100. When this context is provided to a subsequent call to **exportNetworkLink**, the number of exported network links is limited to the first 100 that match the criteria in the given query filter. The web service client may change this **maxResults** attribute of the **WSContext** before any call to the **exportNetworkLink** service to modify the size of the resultant **WSNetworkLink** object array. However, the value specified by the client cannot exceed 100.

Response

The result returned from the **exportNetworkLink** service is an array of **WSNetworkLink** objects matching the selection criteria specified in the query filter. The **WSNetworkLink** can then be modified and/or imported using the **importNetworkLink** API. The format of the **WSNetworkLink** matches that defined by the **importNetworkLink**.

WSNetworkLink

Below is the portion of *Exports.wsdl* that describes **`WSNetworkLink`**, the array of structures returned by **`exportNetworkLink`**. The elements are described in the table that follows.

```
<complexType name="WSNetworkLink">
    <sequence>
     <element name="blockNames" nillable="true" type="impl:ArrayOf_soapenc_string"/>
     <element name="containers" nillable="true" type="impl:ArrayOf_soapenc_string"/>
     <element name="description" nillable="true" type="soapenc:string"/>
     <element name="dhcpOptionSet" nillable="true" type="soapenc:string"/>
     <element name="dhcpPolicySet" nillable="true" type="soapenc:string"/>
     <element name="id" nillable="true" type="soapenc:string"/>
     <element name="name" nillable="true" type="soapenc:string"/>
     <element name="type" nillable="true" type="soapenc:string"/>
    </sequence>
   </complexType>
```

Element	Description and accepted values
blockNames	List of block names for blocks associated with this network link, separated by '|'.
containers	List of full container names corresponding to the blocks listed in blockNames, separated by '|'.
description	A description of the network link
dhcpOptionSet	The name of an Option Set used by subnets associated with this network link.
dhcpPolicySet	The name of a Policy Set used by subnets associated with this network link.
id	Internal id; provide this on a modify request.
name	The name of the network link.
type	The type of the network link, logical or physical.

ExportPrefixPool

Overview

The **exportPrefixPool** API enables the web service client to issue a request to retrieve a list of Prefix Pools from IPAM. This service enables the client to filter the list of Prefix Pools retrieved.

Initialization

Before the **exportPrefixPool** API is called, the web service client *must* call **initExportPrefixPool** to initialize the API. The portion of *Exports.wsdl* that describes the **initExportPrefixPool** request and response messages is shown following.

```
<wsdl:message name="initExportPrefixPoolRequest">
   <wsdl:part name="filter" type="soapenc:string"/>
   <wsdl:part name="options" type="impl:ArrayOf_soapenc_string"/>
</wsdl:message>
<wsdl:message name="initExportPrefixPoolResponse">
   <wsdl:part name="initExportPrefixPoolReturn" type="tns2:WSContext"/>
</wsdl:message>
```

Request

The query string is passed as input from the client to the **initExportPrefixPool** web service. The query string syntax is defined previously. Supported selectors for exporting prefix pools are defined in the following table.

In addition, the **initExportPrefixPool** service accepts the options array, described following the selectors table.

Selector	Description	Example
Name	Exports prefix pool by name.	Name='pool1'
Container	Exports prefix pool by Container name.	Container='Exton' Container='/InControl/USA/Exton' Container ends 'ton' Container begins 'Ext Container contains 'xto'
Net Service	Exports prefix pool by assigned DHCP server .	Netservice='DhcpServer1'
IP Address	Exports the prefix pool that contains that address.	IPAddress='2001:db8::1'
IP Address Range	Exports prefix pool by address range in address/length format.	IPAddressRange='2001 :db8 :0 :1 ::/65'

Response

The response from the **initExportPrefixPool** web service is a **WSContext** object defined previously and *must* be included in each successive call to **exportPrefixPool**, as described below.

Service Invocation

The portion of *Exports.wsdl* that describes the **exportPrefixPool** request and response messages is shown following.

```
<wsdl:message name="endExportPrefixPoolRequest">
   <wsdl:part name="context" type="tns2:WSContext"/>
</wsdl:message>
<wsdl:message name="exportPrefixPoolResponse">
   <wsdl:part name="exportPrefixPoolReturn" type="impl:ArrayOf_tns2_WSPrefixPool"/>
</wsdl:message>
```

Request

The **WSContext** passed as input by the client web service is the **WSContext** object returned by the **initExportPrefixPool** service defined aboveand has the **maxResults** field set to a default value of 100. When this context is provided to a subsequent call to **exportPrefixPool**, the number of exported pools is limited to the first 100, that match the criteria in the given query filter. The web service client may change this **maxResults** attribute of the **WSContext** before any call to the **exportPrefixPool** service to modify the size of the resultant **WSPrefixPool** object array. However, the value specified by the client cannot exceed 100.

Response

The result returned from the **exportPrefixPool** service is an array of **WSPrefixPool** objects matching the selection criteria specified in the query filter. The **WSPrefixPools** can then be modified and/or imported using the **importPrefixPool** API. The format of the **WSPrefixPool** matches that defined by the **importPrefixPool**.

WSPrefixPool

Below is the portion of *Exports.wsdl* that describes **WSPrefixPool**, the array of structures returned by **exportPrefixPool**. The elements are described in the table that follows.

```
<complexType name="WSPrefixPool">
   <sequence>
    <element name="allowClientClasses" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="container" nillable="true" type="soapenc:string"/>
    <element name="delegatedPrefixLength" nillable="true" type="soapenc:int"/>
    <element name="denyClientClasses" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="dhcpOptionSet" nillable="true" type="soapenc:string"/>
    <element name="dhcpPolicySet" nillable="true" type="soapenc:string"/>
    <element name="id" nillable="true" type="soapenc:int"/>
    <element name="length" nillable="true" type="soapenc:int"/>
    <element name="longestPrefixLength" nillable="true" type="soapenc:int"/>
    <element name="name" nillable="true" type="soapenc:string"/>
    <element name="primaryNetService" nillable="true" type="soapenc:string"/>
    <element name="shortestPrefixLength" nillable="true" type="soapenc:int"/>
    <element name="startAddr" nillable="true" type="soapenc:string"/>
    <element name="type" nillable="true" type="soapenc:string"/>
   </sequence>
```

```
</complexType>
```

Element	Description
id	The internal identifier for this address pool object
startAddr	The IP Address of the first address in the pool.
Length	Length of the prefix pool
type	One of "Dynamic PD DHCPv6"," Automatic PD DHCPv6"
name	Address Pool name.
container	The name of the container that holds the block in which the pool is defined.
delegatedPrefixLength	Specifies the default length of the delegated prefix
shortestPrefixLength	Specifies the shortest length of the delegated prefix. CNR only.
longestPrefixLength	Specifies the longest length of the delegated prefix. CNR only.
primaryNet Service	The name of the DHCP server that will serve prefixes from this pool
dhcpOptionSet	The name of an Option Set used with this pool
dhcpPolicySet	The name of a Policy Set used with this pool.
allowClient Classes	An array of Client Classes that are allowed in this address pools. Each element of the array names a different Client Class.
denyClientClasses	An array of Client Classes that are NOT allowed in this address pools. Each array element names a different Client Class.

ExportResourceRecordPendingApproval

Overview

The **exportResourceRecordPendingApproval** API enables the web service client to issue a request to retrieve a list of resource records that are waiting for approval by the invoking administrator. This service enables the client to filter the list of resource records retrieved by requesting administrator, domain name/type and the requested action.

Initialization

Before the **exportResourceRecordPendingApproval** API is called, the web service client *must* call **initExportResourceRecordPendingApproval** to initialize the API. The portion of *Exports.wsdl* that describes the **initExportResourceRecordPendingApproval** request and response messages is shown following.

```
<wsdl:message name="initExportResourceRecordPendingApprovalRequest">
  <wsdl:part name="filter" type="soapenc:string"/>
  <wsdl:part name="options" type="impl:ArrayOf_soapenc_string"/>
</wsdl:message>
<wsdl:message name="initExportResourceRecordPendingApprovalResponse">
  <wsdl:part name="initExportResourceRecordPendingApprovalReturn" type="tns2:WSContext"/>
</wsdl:message>
```

Request

The query string is passed as input from the client to the **initExportResourceRecordPendingApproval** web service in the filter parameter. The query string syntax is defined previously. Supported selectors for exporting device resource records by device are defined in the following table.

Currently, there are no options defined for this service.

Selector	Description	Example
Domain	Select resource records by domain name.	Domain contains "ins.com'
Domain Type	Select resource records by domain type.	Domain Type contains 'Internal'
pendingAction	Select resource records based on the request to "create", "update" or "delete" that resource record	pendingAction='create' pendingAction='delete' pendingAction='update'
adminLoginId	Select resource records by the login id of the administrator requesting the resource record change.	adminLoginId='someuser'

Response

The response from the **initExportResourceRecordPendingApproval** web service is a **WSContext** object defined previously. This **WSContext** object *must* be included in each successive call to **exportResourceRecordPendingApproval**, as described below.

Service Invocation

The portion of *Exports.wsdl* that describes the **exportResourceRecordPendingApproval** request and response messages is shown following.

```
<wsdl:message name="exportResourceRecordPendingApprovalRequest">
  <wsdl:part name="context" type="tns2:WSContext" />
</wsdl:message>
<wsdl:message name="exportResourceRecordPendingApprovalResponse">
  <wsdl:part name="exportResourceRecordPendingApprovalReturn"
    type="impl:ArrayOf_tns2_WSResourceRecPendingApproval"/>
</wsdl:message>
```

Request

The **WSContext** passed as input by the client web service is the **WSContext** object returned by the **initExportResourceRecordPendingApproval** service defined above and has the **maxResults** field set to a default value of 5000. When this context is provided to a subsequent call to **exportResourceRecordPendingApproval**, the number of exported resource records is limited to the first 5000 resource record change requests, or less (see Paging), that match the criteria in the given query filter. The web service client may change this **maxResults** attribute of the **WSContext** before any call to the **exportResourceRecordPendingApproval** service to modify the size of the resultant **WSResourceRecPendingApproval** object array. However, the value specified by the client cannot exceed 5000.

Paging

A resource record change request in IPAM is normalized within the database and thus may be represented by more than a single row in multiple tables. Because of this and for performance, the **exportResourceRecordPendingApproval** service cannot guarantee that the number of **WSResourceRecPendingApproval** objects returned in any single execution of the service will be equal to the max results set on the **WSContext** object. It will, however, always guarantee the number of results to be the max results value or less.

Response

The result returned from the **exportResourceRecordPendingApproval** service is an array of **WSResourceRecPendingApproval** objects matching the selection criteria specified in the query filter. The workflowId returned in **WSResourceRecPendingApproval** can then be used to invoke the **modifyPendingApproval** API.

WSResourceRecPendingApproval

The portion of *Exports.wsdl* that describes **WSResourceRecPendingApproval**, the array of structures returned by **exportResourceRecordPendingApproval** is shown following. The elements are described in the table that follows.

```
<complexType name="WSDeviceResourceRec">
  <sequence>
    <element name="TTL" nillable="true" type="soapenc:string"/>
    <element name="action" nillable="true" type="soapenc:string"/>
    <element name="admin" nillable="true" type="soapenc:string"/>
    <element name="comment" nillable="true" type="soapenc:string"/>
    <element name="container" nillable="true" type="soapenc:string"/>
    <element name="data" nillable="true" type="soapenc:string"/>
    <element name="dateTime" nillable="true" type="soapenc:string"/>
    <element name="domain" nillable="true" type="soapenc:string"/>
    <element name="domainType" nillable="true" type="soapenc:string"/>
    <element name="hostname" nillable="true" type="soapenc:string"/>
    <element name="ipAddress" nillable="true" type="soapenc:string"/>
    <element name="owner" nillable="true" type="soapenc:string"/>
    <element name="resourceRecClass" nillable="true" type="soapenc:string"/>
    <element name="resourceRecType" nillable="true" type="soapenc:string"/>
    <element name="server" nillable="true" type="soapenc:string"/>
    <element name="view" nillable="true" type="soapenc:string"/>
    <element name="workflowId" nillable="true" type="soapenc:int"/>
    <element name="zone" nillable="true" type="soapenc:string"/>
  </sequence>
</complexType>
```

Element	Accepted Values
TTL	The Time To Live for the record.
action	The request – “update”, “create” or “delete”
admin	The login id of the requesting administrator
comment	Comment text associated with the resource record.
container	The name of the container that holds the device for this resource record. This is required only if there is overlapping address space in use and the ip address is in overlapping space. The container is then used to uniquely determine the device.
data	The data portion of the resource record. The format is dependent on the type specified above.
dateTime	The date and time of the approval request.
domain	Domain name of resource record.
domainType	Domain type of resource record.
hostname	The device host name.
ipAddress	The IP Address of the Device.
owner	The owner field of the resource record.
resourceRecClass	The Class of the Resource Record. Defaults to “IN”.
resourceRecordType	The Type of the resource Record.
server	The Time To Live for the record.
view	The name of the view for the domains of this resource record
workflowid	This is required as input to modifyPendingApproval.
zone	The name of the DNS network service as defined in IPAM to import the zone data into.

ExportResourceRecordPendingApprovalStatus

Overview

The **exportResourceRecordPendingApprovalStatus** API enables the web service client to issue a request to retrieve a list of resource records were submitted for approval by the invoking administrator. These updates include those to create, update or delete a resource record.

Initialization

Before the **exportResourceRecordPendingApprovalStatus** API is called, the web service client *must* call **initExportResourceRecordPendingApprovalStatus** to initialize the API. The portion of *Exports.wsdl* that describes the **initExportResourceRecordPendingApprovalStatus** request and response messages is shown following.

```
<wsdl:message name="initExportResourceRecordPendingApprovalRequestStatus">
  <wsdl:part name="filter" type="soapenc:string"/>
  <wsdl:part name="options" type="impl:ArrayOf_soapenc_string"/>
</wsdl:message>
<wsdl:message name="initExportResourceRecordPendingApprovalStatusResponse">
  <wsdl:part name="initExportResourceRecordPendingApprovalStatusReturn" type="tns2:WSContext"/>
</wsdl:message>
```

Request

The query string is passed as input from the client to the **initExportResourceRecordPendingApprovalStatus** web service in the filter parameter. The query string syntax is defined previously. Supported selectors for exporting device resource records by device are defined in the following table.

Currently, there are no options defined for this service.

Selector	Description	Example
Domain	Select resource records by domain name.	Domain contains "ins.com'
Domain Type	Select resource records by domain type.	Domain Type contains 'Internal'
pendingAction	Select resource records based on the request to "create", "update" or "delete" that resource record	pendingAction='create' pendingAction='delete' pendingAction='update'

Response

The response from the **initExportResourceRecordPendingApprovalStatus** web service is a **WSContext** object defined previously. This **WSContext** object *must* be included in each successive call to **exportResourceRecordPendingApprovalStatus** as described below.

Service Invocation

The portion of *Exports.wsdl* that describes the **exportResourceRecordPendingApprovalStatus** request and response messages is shown below.

```
<wsdl:message name="exportResourceRecordPendingApprovalStatusRequest">
  <wsdl:part name="context" type="tns2:WSContext" />
</wsdl:message>
<wsdl:message name="exportResourceRecordPendingApprovalStatusResponse">
  <wsdl:part name="exportResourceRecordPendingApprovalStatusReturn"
    type="impl:ArrayOf_tns2_WSResourceRecPendingApproval"/>
</wsdl:message>
```

Request

The **WSContext** passed as input by the client web service is the **WSContext** object returned by the **initExportResourceRecordPendingApprovalStatus** service defined abov and has the **maxResults** field set to a default value of 5000. When this context is provided to a subsequent call to **exportResourceRecordPendingApprovalStatus**, the number of exported resource records is limited to the first 5000 resource record change requests, or less (see Paging), that match the criteria in the given query filter. The web service client may change this **maxResults** attribute of the **WSContext** before any call to the **exportResourceRecordPendingApprovalStatus** service to modify the size of the resultant **WSResourceRecPendingApproval** object array. However, the value specified by the client cannot exceed 5000.

Paging

A resource record change request in IPAM is normalized within the database and thus may be represented by more than a single row in multiple tables. Because of this and for performance, the **exportResourceRecordPendingApprovalStatus** service cannot guarantee that the number of **WSResourceRecPendingApproval** objects returned in any single execution of the service will be equal to the max results set on the **WSContext** object. It will, however, always guarantee the number of results to be the max results value or less.

Response

The result returned from the **exportResourceRecordPendingApprovalStatus** service is an array of **WSResourceRecPendingApproval** objects matching the selection criteria specified in the query filter.

WSResourceRecPendingApproval

The portion of *Exports.wsdl* that describes **WSResourceRecPendingApproval**, the array of structures returned by **exportResourceRecordPendingApprovalStatus** is shown following. The elements are described in the table that follows.

```
<complexType name="WSDeviceResourceRec">
  <sequence>
    <element name="TTL" nillable="true" type="soapenc:string"/>
    <element name="action" nillable="true" type="soapenc:string"/>
    <element name="admin" nillable="true" type="soapenc:string"/>
    <element name="comment" nillable="true" type="soapenc:string"/>
    <element name="container" nillable="true" type="soapenc:string"/>
    <element name="data" nillable="true" type="soapenc:string"/>
    <element name="dateTime" nillable="true" type="soapenc:string"/>
    <element name="domain" nillable="true" type="soapenc:string"/>
    <element name="domainType" nillable="true" type="soapenc:string"/>
    <element name="hostname" nillable="true" type="soapenc:string"/>
    <element name="ipAddress" nillable="true" type="soapenc:string"/>
    <element name="owner" nillable="true" type="soapenc:string"/>
    <element name="resourceRecClass" nillable="true" type="soapenc:string"/>
    <element name="resourceRecType" nillable="true" type="soapenc:string"/>
    <element name="server" nillable="true" type="soapenc:string"/>
    <element name="view" nillable="true" type="soapenc:string"/>
    <element name="workflowId" nillable="true" type="soapenc:int"/>
    <element name="zone" nillable="true" type="soapenc:string"/>
  </sequence>
</complexType>
```

Element	Accepted Values
TTL	The Time To Live for the record.
action	The request – “update”, “create” or “delete”
admin	The login id of the requesting administrator
comment	Comment text associated with the resource record.
container	The name of the container that holds the device for this resource record. This is required only if there is overlapping address space in use and the ip address is in overlapping space. The container is then used to uniquely determine the device.
data	The data portion of the resource record. The format is dependent on the type specified above.
dateTime	The date and time of the approval request.
domain	Domain name of resource record.
domainType	Domain type of resource record.
hostname	The device host name.
ipAddress	The IP Address of the Device.
owner	The owner field of the resource record.
resourceRecClass	The Class of the Resource Record. Defaults to “IN”.
resourceRecordType	The Type of the resource Record.
server	The Time To Live for the record.
view	The name of the view for the domains of this resource record
workflowid	This is required as input to modifyPendingApproval.
zone	The name of the DNS network service as defined in IPAM to import the zone data into.

ExportResourceRecordRestoreList

Overview

The **exportResourceRecordRestoreList** API enables the web service client to issue a request to retrieve a list of resource records that have been deleted and may be eligible for restoring. This service enables the client to filter the list of resource records exported. A superuser may filter the results based on the requesting administrator(not available for non-superusers). Resource records deleted from a device may be filtered based on hostname or IP Address of the device. Resource records deleted from a zone may be filtered based on the zone, server or view.

Initialization

Before the **exportResourceRecordRestoreList** API is called, the web service client *must* call **initExportResourceRecordRestoreList** to initialize the API. The portion of *Exports.wsdl* that describes the **initExportResourceRecordRestoreList** request and response messages is shown following.

```
<wsdl:message name="initExportResourceRecordRestoreListRequest">
      <wsdl:part name="filter" type="soapenc:string"/>
      <wsdl:part name="options" type="impl:ArrayOf_soapenc_string"/>
</wsdl:message>
 <wsdl:message name="initExportResourceRecordRestoreListResponse">
      <wsdl:part name="initExportResourceRecordRestoreListReturn" type="tns2:WSContext"/
</wsdl:message>
```

Request

The query string is passed as input from the client to the **initExportResourceRecordRecordList** web service in the filter parameter. The query string syntax is defined previously. Supported selectors for exporting device resource records by device are defined in the following table.

Currently, there are no options defined for this service.

Selector	Description	Example
Admin	Export resource records deleted by a particular user. This filter is only available for superusers.	Admin='someuser' Admin ends 'ser' Admin begins 'some' Admin contains 'ome'
Name	Export resource records deleted from a device with a particular hostname.	Name ='host; Name ends 'ost' Name begins 'hos' Name contains 'os'
IPAddress	Export resource records deleted from a device with the given IP Address.	IPAddress='168.0.0.1'
Domain	Export deleted resource records tied to a given domain.	Domain='ins.com.' Domain begins 'ins.c' Domain ends '.com.'

Selector	Description	Example
		Domain contains 's.com'
DomainType	Export deleted resource records tied to a given domain type	DomainType='Internal' DomainType ends 'nal' DomainType begins 'Int' DomainType contains 'ter'
`RR_type`	Export deleted resource records of a particular type.	`RR_type` ='CNAME' `RR_type` ends 'NAME' `RR_type` begins 'C' `RR_type` contains 'AM'
Owner	Export deleted resource records with a particular Owner field.	Owner='XYZ' Owner ends 'Z' Owner begins 'X' Owner contains 'Y'
RDATA	Export deleted resource records with a particular RDATA field.	RDATA='XYZ' RDATA ends 'Z' RDATA begins 'X' RDATA contains 'Y'

Response

The response from the **`initExportResourceRecordRestoreList`** web service is a **`WSContext`** object defined previously. This **`WSContext`** object *must* be included in each successive call to **`exportResourceRecordRestoreList`** , as described below.

Service Invocation

The portion of *Exports.wsdl* that describes the **`exportResourceRecordRestoreList`** request and response messages is shown following.

```
<wsdl:message name="exportResourceRecordRestoreListRequest">
   <wsdl:part name="context" type="tns2:WSContext"/>
</wsdl:message>
<wsdl:message name="exportResourceRecordRestoreListResponse">
   <wsdl:part name="exportResourceRecordRestoreListReturn"
   type="impl:ArrayOf_tns2_WSRestoreResourceRecord"/>
</wsdl:message>
```

Request

The **`WSContext`** passed as input by the client web service is the **`WSContext`** object returned by the **`initExportResourceRecordRestoreList`** service defined above and has the **`maxResults`** field set to a default value of 5000. When this context is provided to a subsequent call to **`exportResourceRecordRestoreList`**, the number of exported resource records is limited to the first 5000 resource record change requests, or less (see Paging), that match the criteria in the given query filter. The web service client may change this **`maxResults`** attribute of the **`WSContext`** before any call to the **`exportResourceRecordRestoreList`** service to modify the size of the resultant

WSRestoreResourceRecord object array. However, the value specified by the client cannot exceed 5000.

Paging

A resource record change request in IPAM is normalized within the database and thus may be represented by more than a single row in multiple tables. Because of this and for performance, the **exportResourceRecordRestoreList** service cannot guarantee that the number of **WSRestoreResourceRecord** objects returned in any single execution of the service will be equal to the max results set on the **WSContext** object. It will, however, always guarantee the number of results to be the max results value or less.

Response

The result returned from the **exportResourceRecordRestoreList** service is an array of **WSRestoreResourceRecord** objects matching the selection criteria specified in the query filter. **WSRestoreResourceRecord** thus returned can then be used to invoke the **restoreDeletedResourceRecord** API.

WSRestoreResourceRecord

The portion of *Exports.wsdl* that describes **WSRestoreResourceRecord**, the array of structures returned by **exportResourceRecordRestoreList** is shown following. The elements are described in the table that follows.

```
<complexType name="WSRestoreResourceRecord">
    <sequence>
      <element name="TTL" nillable="true" type="soapenc:string"/>
      <element name="admin" nillable="true" type="soapenc:string"/>
      <element name="comment" nillable="true" type="soapenc:string"/>
      <element name="data" nillable="true" type="soapenc:string"/>
      <element name="dateTime" nillable="true" type="soapenc:string"/>
      <element name="domain" nillable="true" type="soapenc:string"/>
      <element name="domainType" nillable="true" type="soapenc:string"/>
      <element name="hostname" nillable="true" type="soapenc:string"/>
      <element name="ipAddress" nillable="true" type="soapenc:string"/>
      <element name="owner" nillable="true" type="soapenc:string"/>
      <element name="resourceRecClass" nillable="true" type="soapenc:string"/>
      <element name="resourceRecType" nillable="true" type="soapenc:string"/>
      <element name="restoreId" nillable="true" type="soapenc:int"/>
      <element name="ignoreDuplicateOwnerWarning" nillable="true" type="soapenc:string"/>
     </sequence>
</complexType>
```

Element	Accepted Values
TTL	The Time To Live for the record.
admin	The login id of the administrator who deleted the record.
comment	Comment text associated with the resource record.
data	The data portion of the resource record. The format is dependent on the type specified.
dateTime	The date and time when the record was deleted.
domain	Domain name of resource record.
domainType	Domain type of resource record.
hostname	The device host name.
ipAddress	The IP Address of the Device.
owner	The owner field of the resource record.
resourceRecClass	The Class of the Resource Record. Defaults to "IN".
resourceRecordType	The Type of the resource Record.
server	The name of the DNS network service as defined in IPAM to import the zone data into for a zone resource record.
view	The name of the view for the domains of a zone resource record
zone	The name of the zone from which the zone resource record was deleted.
restoreId	Id that is used by IPAM internally to identify the resource record to be restored.
ignoreDuplicateOwnerWarning	This true/false flag indicates if the restore should ignore duplicate owner warning.

Updates

This section explains the web services available for updating information in IPAM.

UseNextReservedIPAddress

Overview

The **UseNextReservedIPAddress** API enables the web service client to mark the next reserved IP Address in the specified block, for the specified device type, to in-use. The block must have a status if "In Use/Deployed". Within this block, there should be a range of addresses with a type of "Reserved" and a status of "reserved" for the given device type. The next lowest IP address within the range will be assigned a type of "Static" and a status of "in-use". If a hostname is specified, it will be applied to the device associated with the IP Address. In addition, there is an option to add resource records for the device.

This service is available as an operation in the **IncUseNextReservedIPAddress** web service. You can see the complete WSDL at:

http://localhost:8080/inc-ws/services/IncUseNextReservedIPAddress?wsdl

Request and Response Messages

Below is the portion of *IncUseNextReservedIPAddress.wsdl* that describes the **UseNextReservedIPAddress** request and response messages.

```
<wsdl:message name="useNextReservedIPAddressRequest">
  <wsdl:part name="inpDevice" type="tns1:WSDevice" />
</wsdl:message>
<wsdl:message name="useNextReservedIPAddressResponse">
  <wsdl:part name="useNextReservedIPAddressReturn" type="soapenc:string" />
</wsdl:message>
```

Response

The string that is returned contains the IP Address used to satisfy the request.

Request

The complex type **WSDevice**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSDevice

The portion of *IncUseNextReservedIPAddress.wsdl* that describes **WSDevice**, the parameter structure passed to **UseNextReservedIPAddress**, is shown following. The elements are described in the table that follows.

```
<complexType name="WSDevice">
  <sequence>
    <element name="view" nillable="true" type="soapenc:string" />
    <element name="hwType" nillable="true" type="soapenc:string" />
    <element name="addressType" nillable="true" type="soapenc:string" />
    <element name="ipAddress" nillable="true" type="soapenc:string" />
```

```
   <element name="resourceRecordFlag" nillable="true" type="soapenc:string" />
   <element name="MACAddress" nillable="true" type="soapenc:string" />
   <element name="deviceType" nillable="true" type="soapenc:string" />
   <element name="domainName" nillable="true" type="soapenc:string" />
   <element name="container" nillable="true" type="soapenc:string" />
   <element name="description" nillable="true" type="soapenc:string" />
   <element name="hostname" nillable="true" type="soapenc:string" />
   <element name="aliases" nillable="true" type="impl:ArrayOf soapenc string" />
   <element name="userDefinedFields" nillable="true" type="impl:ArrayOf_soapenc_string" />
 </sequence>
</complexType>
```

Element	Accepted Values	Required	Return Code	Fault String
view	Not used			
hwType	Not used			
addressType	Not used			
ipAddress	The IP Address of the block	Yes	-26	Invalid Ip Address: *address*
resourceRecordFlag	Whether or not to add resource records for this device. Accepted values are **true** or **false**. If not specified, defaults to false.	No		
MACAddress	Not used			
deviceType	The device type associated with the reserved IP address.	Yes	-47	Device type not found: *type*
domainName	Not used			
container	Not used			
description	Not used			
hostname	Valid host name or **APPLYNAMINGPOLICY**.	Yes		
aliases	Not used			
userDefinedFields	Not used			

Other returnCodes and faultstrings

Return Code	Faultstring
-1	Error saving resource records: *error* (system errors)
-5	Container not found for block
-23	No matching in-use blocks found
-36	Block not found
-47	Error creating DnsResourceRecHelper (system error)
-61	Forward domain not found for: *address*
-62	Subnet not found for: *address*

Deletes

The Delete APIs allow a client program to delete objects in the system. Each of these services is available as an operation in the Deletes web service. You can see the complete WSDL at:

http://localhost:8080/inc-ws/services/Deletes?wsdl

DeleteAddrPool

Overview

The **`DeleteAddrPool`** API enables the web service client to delete an Address Pool from a block . By specifying the address pool to be deleted, the web service will validate and delete the address pool.

Request and Response Messages

The portion of *Deletes.wsdl* that describes the **`deleteAddrPool`** request and response messages is shown following.

```
<wsdl:operation name="deleteAddrPool" parameterOrder="inpAddrPool deleteDevicesInAddrpool">
  <wsdl:input message="impl:deleteAddrPoolRequest" name="deleteAddrPoolRequest"/>
  <wsdl:output message="impl:deleteAddrPoolResponse" name="deleteAddrPoolResponse"/>
</wsdl:operation>
<wsdl:message name="deleteAddrPoolRequest ">
  <wsdl:part name="inpAddrPool" type="tns2:WSAddrpool"/>
  <wsdl:part name="deleteDevicesInAddrpool" type="xsd:boolean"/>
</wsdl:message>
<wsdl:message name="deleteAddrPoolResponse">
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **`WSAddrpool`**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSAddrpool

The portion of *Deletes.wsdl* that describes **`WSAddrpool`**, the parameter structure passed to DeleteAddrPool, is shown following. The elements are described in the table that follows.

```
<complexType name="WSAddrpool">
   <sequence>
<element name="allowClientClasses" nillable="true" type="impl:ArrayOf_soapenc_string"/>
<element name="container" nillable="true" type="soapenc:string"/>
<element name="denyClientClasses" nillable="true" type="impl:ArrayOf_soapenc_string"/>
<element name="dhcpOptionSet" nillable="true" type="soapenc:string"/>
<element name="dhcpPolicySet" nillable="true" type="soapenc:string"/>
<element name="endAddr" nillable="true" type="soapenc:string"/>
<element name="failoverNetService" nillable="true" type="soapenc:string"/>
<element name="id" nillable="true" type="soapenc:string"/>
<element name="name" nillable="true" type="soapenc:string"/>
<element name="primaryNetService" nillable="true" type="soapenc:string"/>
<element name="sharename" nillable="true" type="soapenc:string"/>
<element name="startAddr" nillable="true" type="soapenc:string"/>
<element name="type" nillable="true" type="soapenc:string"/>
<element name="prefixLength" nillable="true" type="soapenc:string"/>
  </sequence>
</complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
Name	The name of the address pool to be deleted.	Yes	-117	Address Pool not found
Startaddr	The start Address of the address pool to be deleted.	Yes	-117	Address Pool not found
Container	The container in which the address pool resides in. This is to disambiguate the address pool.	Yes	-117	Address Pool not found

DeleteAdmin

Overview

The **deleteAdmin** API enables the web service client to delete administrators from IPAM . By specifying the Login ID, the web service will identify and delete the administrator.

Request and Response Messages

The portion of *Deletes.wsdl* that describes the **deleteAdmin** request and response messages is shown following.

```
<wsdl:message name="deleteAdminResponse">
</wsdl:message>
<wsdl:message name="deleteAdminRequest">
      <wsdl:part name="inpAdmin" type="tns2:WSAdmin"/>
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSAdmin**, which is passed as input from the client to the web service, matches that defined by **importAdmin**.

Parameters

WSAdmin

The portion of *Deletes.wsdl* that describes **WSAdmin**, the parameter structure passed to **deleteAdmin**, is shown in the **importAdmin** section. The required elements are described in the table that follows. The other elements are ignored.

Element	Accepted Values	Required	Return Code	Faultstring
loginId	The login id of the administrator to be deleted.	Yes	-292 -293 -295 -309	Admin is not authorized to access Administrator functions Administrator login ID is required Administrator not found for login: *loginid* Delete disallowed for id 0: *loginid* Delete disallowed for invoking user: *loginid*

DeleteAdmin

Element	Accepted Values	Required	Return Code	Faultstring
				Delete disallowed: normal admin cannot delete a master admin

DeleteAdminRole

Overview

The **deleteAdminRole** API enables the web service client to delete administrator roles from IPAM . By specifying the name of the administrator role to be deleted, the web service will identify and delete the administrator role.

Request and Response Messages

The portion of *Deletes.wsdl* that describes the **deleteAdminRole** request and response messages is shown following.

```
<wsdl:message name="deleteAdminRoleResponse">
</wsdl:message>
<wsdl:message name="deleteAdminRoleRequest">
      <wsdl:part name="inpAdminRole" type="tns2:WSAdminRole"/>
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSAdminRole**, which is passed as input from the client to the web service matches that defined by the **importAdminRole**.

.

Parameters

WSAdminRole

The portion of *Deletes.wsdl* that describes **WSAdminRole**, the parameter structure passed to **deleteAdminRole**, is shown in the **importAdmin** section. The required elements are described in the table that follows. The other elements are ignored.

Element	Accepted Values	Required	Return Code	Faultstring
roleName	The name of the administrator role to be deleted.	Yes	-300 -303 -308 -310	Administrator role name is required Administrator role not found for name: *name* Delete of hidden role not allowed: *name* Delete of role in use not allowed: *name*

DeleteAggregateBlock

Overview

The **DeleteAggregateBlock** API enables the web service client to delete an intermediate level Aggregate block from the block hierarchy. By specifying the block to be deleted, the web service will validate and delete the block. It will also adjust the parent block assignments of any child blocks.

Request and Response Messages

The portion of *Deletes.wsdl* that describes the **deleteAggregate** request and response messages is shown following.

```
<wsdl:operation name="deleteAggregateBlock" parameterOrder="inpBlock">
  <wsdl:input message="impl:deleteAggregateBlockRequest"
    name="deleteAggregateBlockRequest"/>
  <wsdl:output message="impl:deleteAggregateBlockResponse"
    name="deleteAggregateBlockResponse"/>
</wsdl:operation>
<wsdl:message name="deleteAggregateBlockRequest ">
    <wsdl:part name="inpBlock" type="tns2:WSBlock4Delete"/>
</wsdl:message>
<wsdl:message name="deleteAggregateBlockResponse">
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSBlock4Delete**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSBlock4Delete

The portion of *Deletes.wsdl* that describes **WSBlock4Delete**, the parameter structure passed to DeleteAggregateBlock, is shown following. The elements are described in the table that follows.

```
<complexType name="WSBlock4Delete">
  <sequence>
    <element name="blockAddr" nillable="true" type="soapenc:string"/>
    <element name="blockSize" nillable="true" type="soapenc:int"/>
    <element name="container" nillable="true" type="soapenc:string"/>
  </sequence>
</complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
blockAddr	The start address of the aggregate block to be deleted.	Yes	-127	Block start and parent block

Element	Accepted Values	Required	Return Code	Faultstring
				address both required
			-12	Could not convert <address> to ipAddress
blockSize	The size of the block in short-notation (e.g., 24 for a 255.255.255.0 network).	Yes	-15	Block size invalid: <size>
container	The name of the container from which to delete the aggregate block. Names can be in either short or long format. Short format example: Dallas. Long format example: InControl/Texas/Dallas. Long format eliminates ambiguity in cases where there are duplicate container names.	Yes	-42	Missing container name
			-5	Could not find container: <container>
			-99	Admin is not authorized to add blocks to this container

DeleteBlock

Overview

The **DeleteBlock** API enables the web service client to delete blocks from IPAM.

Request and Response Messages

The portion of *Deletes.wsdl* that describes the **DeleteBlock** request and response messages is shown following.

```
<wsdl:operation name="deleteBlock" parameterOrder="container blockName">
  <wsdl:input message="impl:deleteBlockRequest" name="deleteBlockRequest"/>
  <wsdl:output message="impl:deleteBlockResponse" name="deleteBlockResponse"/>
</wsdl:operation>

<wsdl:message name="deleteBlockRequest">
  <wsdl:part name="container" type="soapenc:string"/>
  <wsdl:part name="blockName" type="soapenc:string"/>
</wsdl:message>

<wsdl:message name="deleteBlockResponse">
</wsdl:message>
```

Response

There is no data in the response.

Request

The request takes two parameters, block name and container name. Their use is described in more detail in the next section.

Parameters

Element	Accepted Values	Required	Return Code	Faultstring
blockName	The name of an existing block. This is often the address followed by CIDR size, e.g. 10.0.0.0/8. However, if the block name was changed during allocation, then the modified value should be supplied here.	Yes	-36 -97 -98 -100 -101 -102 -105	Block not found Block not unique Parent not found. Cannot delete free aggregate. Error deleting block IP Address Cleanup Error Error reclaiming blocks
Container	The name of the container holding the block. This is required if overlapping space is in use and the block overlaps	No, unless block is overlappe	-99	Container not found

Element	Accepted Values	Required	Return Code	Faultstring
	another in the system.	d.		

DeleteContainer

Overview

The **DeleteContainer** API enables the web service client to delete containers from IPAM.

Request and Response Messages

The portion of *Deletes.wsdl* that describes the **deleteContainer** request and response messages is shown following.

```
<wsdl:operation name="deleteContainer" parameterOrder="fullName">
  <wsdl:input message="impl:deleteContainerRequest" name="deleteContainerRequest"/>
  <wsdl:output message="impl:deleteContainerResponse" name="deleteContainerResponse"/>
</wsdl:operation>
<wsdl:message name="deleteContainerRequest
  <wsdl:part name="fullName" type="soapenc:string"/>
</wsdl:message>
<wsdl:message name="deleteContainerResponse">
</wsdl:message>
```

Response

There is no data in the response.

Request

The request contains a single string parameter, **fullName**.

Parameters

fullName

This is name of the container. The name can be either qualified or unqualified, but must be unique. A qualified name must start with the root container and include the complete container path to the desired container. The container names should be separated by slashes.

Other returnCodes and faultstrings

ReturnCode	Faultstring
-2	Container delete failed (database error)
-5	Container not found
-99	Access denied
-132	Invalid Container name supplied
-143	Container name ambiguous
-144	Container has children
-145	Container has blocks
-146	Container has services

DeleteDevice

Overview

The **DeleteDevice** API enables the web service client to delete devices from IPAM.

Note that this is not used to delete devices of type "Interface". Use the ModifyBlock API to delete Interface-type devices.

Request and Response Messages

The portion of **Deletes.wsdl** that describes the **deleteDevice** request and response messages is shown following.

```
<wsdl:operation name="deleteDevice" parameterOrder="inpDev">
  <wsdl:input message="impl:deleteDeviceRequest" name="deleteDeviceRequest"/>
  <wsdl:output message="impl:deleteDeviceResponse" name="deleteDeviceResponse"/>
</wsdl:operation>
<wsdl:message name="deleteDeviceRequest">
  <wsdl:part name="inpDev" type="tns2:WSDevice"/>
</wsdl:message>
<wsdl:message name="deleteDeviceResponse">
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSDevice**, which is passed as input from the client to the web service, is described in the next section. For consistency, this is the same structure that is passed to **ImportDevice**. However, only two of the fields are used.

Parameters

WSDevice

The portion of *Deletes.wsdl* that describes **WSDevice**, the parameter structure passed to **DeleteDevice**, is shown following. The required elements are described in the table that follows. The other elements are ignored.

```
<complexType name="WSDevice">
  <sequence>
    <element name="MACAddress" nillable="true" type="soapenc:string"/>
    <element name="addressType" nillable="true" type="soapenc:string"/>
    <element name="aliases" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="container" nillable="true" type="soapenc:string"/>
    <element name="description" nillable="true" type="soapenc:string"/>
    <element name="deviceType" nillable="true" type="soapenc:string"/>
    <element name="domainName" nillable="true" type="soapenc:string"/>
    <element name="domainType" nillable="true" type="soapenc:string"/>
    <element name="dupWarning" nillable="true" type="soapenc:string"/>
    <element name="hostname" nillable="true" type="soapenc:string"/>
    <element name="hwType" nillable="true" type="soapenc:string"/>
    <element name="ipAddress" nillable="true" type="soapenc:string"/>
    <element name="resourceRecordFlag" nillable="true" type="soapenc:string"/>
    <element name="userDefinedFields" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element maxOccurs="unbounded" name="interfaces" nillable="true" type="tns2:WSInterface"/>
    <element name="id" nillable="true" type="soapenc:int"/>
```

```
    <element name="excludeFromDiscovery" nillable="true" type="soapenc:string"/
  </sequence>
</complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
ipAddress	The IP Address of the device	Yes	-26 -27 -28	Invalid IP Address: *address* IP Address not found IP Address ambiguous
container	The name of the container that contains the device.	Yes, if overlapping space is in use and the device is in an overlapped block.	-28	IP Address ambiguous

DeleteDeviceInterface

Overview

The **DeleteDeviceInterface** API enables the web service client to delete device interfaces from IPAM.

Note that this is not used to delete devices of type "Interface". Use the ModifyBlock API to delete Interface-type devices.

Request and Response Messages

The portion of *Deletes.wsdl* that describes the **deleteDeviceInterface** request and response messages is shown following.

```
<wsdl:operation name="deleteDeviceInterface" parameterOrder="inpDev">
  <wsdl:input message=" "impl:deleteDeviceInterfaceRequest"
    name="deleteDeviceInterfaceRequest""/>
  <wsdl:output message=" impl:deleteDeviceInterfaceResponse"
    name="deleteDeviceInterfaceResponse"/>
</wsdl:operation>
<wsdl:message name="deleteDeviceInterfaceRequest">
  <wsdl:part name="inpDev" type="tns2:WSDevice"/>
</wsdl:message>
<wsdl:message name="deleteDeviceInterfaceResponse">
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex types **WSDevice** and **WSInterface**, which are passed as input from the client to the web service, are described in the next section. For consistency, these are the same structures that are passed to **ImportDevice**. However, only three of the fields are used.

Parameters

WSDevice

The portion of *Deletes.wsdl* that describes **WSDevice** and **WSInterface**, the parameter structures passed to **DeleteDeviceInterface**, are shown following. The elements are described in the table that follows.

```
<complexType name="WSDevice">
  <sequence>
    <element name="MACAddress" nillable="true" type="soapenc:string"/>
    <element name="addressType" nillable="true" type="soapenc:string"/>
    <element name="aliases" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="container" nillable="true" type="soapenc:string"/>
    <element name="description" nillable="true" type="soapenc:string"/>
    <element name="deviceType" nillable="true" type="soapenc:string"/>
    <element name="domainName" nillable="true" type="soapenc:string"/>
    <element name="domainType" nillable="true" type="soapenc:string"/>
    <element name="dupWarning" nillable="true" type="soapenc:string"/>
    <element name="hostname" nillable="true" type="soapenc:string"/>
    <element name="hwType" nillable="true" type="soapenc:string"/>
    <element name="ipAddress" nillable="true" type="soapenc:string"/>
    <element name="resourceRecordFlag" nillable="true" type="soapenc:string"/>
```

```
    <element name="userDefinedFields" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element maxOccurs="unbounded" name="interfaces" nillable="true" type="tns2:WSInterface"/>
    <element name="id" nillable="true" type="soapenc:int"/>
    <element name="excludeFromDiscovery" nillable="true" type="soapenc:string"/>
  </sequence>
</complexType>
<complexType name="WSNetElementInterface">
  <sequence>
    <element name="id" nillable="true" type="soapenc:int"/>
    <element name="interfaceName" nillable="true" type="soapenc:string"/>
    <element name="netElementName" nillable="true" type="soapenc:string"/>
    <element name="status" nillable="true" type="soapenc:string"/>
  </sequence>
</complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
MACAddress	Ignored	No		
addressType	Ignored	No		
aliases	Ignored	No		
container	The name of the container that contains the device.	Yes, if overlapping space is in use and the device is in an overlapped block.	-28	IP Address ambiguous
description	Ignored	No		
deviceType	Ignored	No		
domainName	Ignored	No		
domainType	Ignored	No		
dupWarning	Ignored	No		
hostname	Ignored	No		
hwType	Ignored	No		
ipAddress	The IP Address of the device	Yes	-26 -27 -28 -127	Invalid IP Address: *address* IP Address not found IP Address ambiguous IP Address is required
resourceRecord Flag	Ignored	No		
userDefined Fields	Ignored	No		

Element	Accepted Values	Required	Return Code	Faultstring
interfaces	An array of WSInterface structures. Each element in the array corresponds to one interface to be deleted. The fields in the WSInterface structure are: name: Interface Name (Required) The following are ignored: - addressType - container - excludeFromDiscovery - hwType - id - ipAddress - macAddress - sequence - virtual	Yes	-20 -19 -38	Must specify interface name Interface not found for device: *ipAddress* with interface name: *name* Cannot delete all interfaces. Use DeleteDevice.
id	Ignored	No		
excludeFrom Discovery	Ignored	No		

DeleteDeviceResourceRecord

Overview

The **DeleteDeviceResourceRecord** API enables the web service client to delete resource records from IPAM that are affiliated with a device.

Request and Response Messages

The portion of *Deletes.wsdl* that describes the **deleteDeviceResourceRecord** request and response messages is shown following.

```
<wsdl:operation name="deleteDeviceResourceRecord" parameterOrder="inpRR">
  <wsdl:input message="impl:deleteDeviceResourceRecordRequest"
    name="deleteDeviceResourceRecordRequest"/>
  <wsdl:output message="impl:deleteDeviceResourceRecordResponse"
    name="deleteDeviceResourceRecordResponse"/>
</wsdl:operation>
<wsdl:message name="deleteDeviceResourceRecordRequest">
  <wsdl:part name="inpRR" type="tns2:WSDeviceResourceRec"/>
</wsdl:message>
<wsdl:message name="deleteDeviceResourceRecordResponse">
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSDeviceResourceRec**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSDeviceResourceRec

The portion of *Deletes.wsdl* that describes **WSDeviceResourceRec**, the parameter structure passed to **DeleteDeviceResourceRecord**, is shown following. The elements are described in the table that follows.

```
<complexType name="WSDeviceResourceRec">
  <sequence>
    <element name="TTL" nillable="true" type="soapenc:string"/>
    <element name="comment" nillable="true" type="soapenc:string"/>
    <element name="container" nillable="true" type="soapenc:string"/>
    <element name="data" nillable="true" type="soapenc:string"/>
    <element name="domain" nillable="true" type="soapenc:string"/>
    <element name="domainType" nillable="true" type="soapenc:string"/>
    <element name="hostname" nillable="true" type="soapenc:string"/>
    <element name="id" nillable="true" type="soapenc:int"/>
    <element name="ipAddress" nillable="true" type="soapenc:string"/>
    <element name="owner" nillable="true" type="soapenc:string"/>
    <element name="resourceRecClass" nillable="true" type="soapenc:string"/>
    <element name="resourceRecType" nillable="true" type="soapenc:string"/>
  </sequence>
</complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
TTL	Time To Live	No		
comment	Ignored			
container	The name of the container that holds the device. This is required only if there is overlapping address space in use and the ip address is in overlapping space. The container is then used to uniquely determine the device.	Only if ipAddress in overlapping space		
data	The RData portion of the record to delete. This *must* match the RData exactly.	Yes	-91	Data field required
domain	The name of the domain for this resource record	Yes	-60 -135	Domain not found: *domain* Domain required: *domain*
domainType	The domain type of the domain. Defaults to "Default".	No	-81	Domain type not found: *domainType*
hostname	The device host name.	Yes, unless ipAddress is specified	-133 -121 -120	Hostname or ip address required Hostname not unique: *hostname* No device with hostname: *hostname*
id	Ignored			
ipAddress	The IP Address of the device	Yes, unless host name is specified	-133 -26 -28 -120	Hostname or ip address required Invalid IP Address: *address* Specify container. IP Address not unique: *address* No device with ipaddress: *address*
Owner	The ow ner field of the record to be deleted. This must match exactly.	Yes	-89 -134	Owner field required Invalid character in Owner *owner*
resourceRecClass	The Resource Record class. This defaults to "IN"	No		
resourceRecType	The type of resource record being deleted	Yes	-90 -92	Type field required Unknown type: *type*

Other returnCodes and faultstrings

ReturnCode	Faultstring
-2	Exception reading domain, device or resource record (database error)
-124	DNS Resource record not found
-125	DNS Resource record not unique
401	<401>Unauthorized

DeleteDomain

Overview

The **DeleteDomain** API enables the web service client to delete domains from IPAM.

Request and Response Messages

The portion of *Deletes.wsdl* that describes the **deleteDomain** request and response messages is shown following.

```
<wsdl:message name="deleteDomainRequest">
  <wsdl:part name="inpDomain" type="tns2:WSDomain"/>
</wsdl:message>
<wsdl:message name="deleteDomainResponse">
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSDomain**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSDomain

The portion of *Deletes.wsdl* that describes **WSDomain**, the parameter structure passed to **DeleteDomain**, is shown following. The elements are described in the table that follows.

```
<complexType name="WSDomain">
  <sequence>
    <element name="contact" nillable="true" type="soapenc:string"/>
    <element name="defaultTTL" nillable="true" type="soapenc:string"/>
    <element name="delegated" type="xsd:boolean"/>
    <element name="derivative" nillable="true" type="soapenc:string"/>
    <element name="domainName" nillable="true" type="soapenc:string"/>
    <element name="domainType" nillable="true" type="soapenc:string"/>
    <element name="expire" nillable="true" type="soapenc:string"/>
    <element name="id" nillable="true" type="soapenc:string"/>
    <element name="infoTemplate" nillable="true" type="soapenc:string"/>
    <element name="managed" type="xsd:boolean"/>
    <element name="negativeCacheTTL" nillable="true" type="soapenc:string"/>
    <element name="refresh" nillable="true" type="soapenc:string"/>
    <element name="retry" nillable="true" type="soapenc:string"/>
    <element name="reverse" type="xsd:boolean"/>
    <element name="serialNumber" nillable="true" type="soapenc:long"/>
    <element name="templateDomain" nillable="true" type="soapenc:string"/>
    <element name="userDefinedFields" nillable="true" type="impl:ArrayOf_soapenc_string"/>
  </sequence>
</complexType>
```

Element	Description and accepted values	Required	Return Code	Faultstring
contact	Ignored	No		
defaultTTL	Ignored	No		
delegated	Ignored	No		
derivitive	Ignored	No		
domainName	The name of the domain to delete.	Yes	-60 -95 -135 -232 -233	Domain not found: *domainType*/*domain* Cannot delete . domain from Default domain type. Domain required Cannot delete *domain* because it is a parent domain. Could not delete domain *domain*: zones may be attached to it.
domainType	Domain type name already defined to IPAM. If not specified, the "Default" domain type will be used.	Yes, when not "Default".	-81	Domain type not found: *domainType*
expire	Ignored	No		
id	Ignored	No		
infoTemplate	Ignored	No		
managed	Ignored	No		
negativeCacheTTL	Ignored	No		
refresh	Ignored	No		
retry	Ignored	No		
reverse	Ignored	No		
serialNumber	Ignored	No		
templateDomain	Ignored	No		
userDefinedFields	Ignored	No		

Other returnCodes and faultstrings

ReturnCode	Faultstring
-2	Exception reading domain or domain type record (database error)
-99	Access denied

DeleteDomainResourceRecord

Overview

The **DeleteDomainResourceRecord** API enables the web service client to delete resource records from IPAM that are affiliated with a domain.

Request and Response Messages

The portion of *Deletes.wsdl* that describes the **deleteDomainResourceRecord** request and response messages is shown following.

```
<wsdl:operation name="deleteDomainResourceRecord" parameterOrder="inpRR">
  <wsdl:input message="impl:deleteDomainResourceRecordRequest"
    name="deleteDomainResourceRecordRequest"/>
  <wsdl:output message="impl:deleteDomainResourceRecordResponse"
    name="deleteDomainResourceRecordResponse"/>
</wsdl:operation>
<wsdl:message name="deleteDomainResourceRecordRequest">
  <wsdl:part name="inpRR" type="tns2:WSDomainResourceRec"/>
</wsdl:message>
<wsdl:message name="deleteDomainResourceRecordResponse">
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSDomainResourceRec**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSDomainResourceRecord

The portion of *Deletes.wsdl* that describes **WSDomainResourceRec**, the parameter structure passed to **DeleteDomainResourceRecord**, is shown following. The elements are described in the table that follows.

```
<complexType name="WSDeviceResourceRec">
  <sequence>
    <element name="TTL" nillable="true" type="soapenc:string"/>
    <element name="comment" nillable="true" type="soapenc:string"/>
    <element name="data" nillable="true" type="soapenc:string"/>
    <element name="deviceRecFlag" type="xsd:boolean"/>
    <element name="domain" nillable="true" type="soapenc:string"/>
    <element name="domainType" nillable="true" type="soapenc:string"/>
    <element name="id" nillable="true" type="soapenc:int"/>
    <element name="owner" nillable="true" type="soapenc:string"/>
    <element name="resourceRecClass" nillable="true" type="soapenc:string"/>
    <element name="resourceRecType" nillable="true" type="soapenc:string"/>
  </sequence>
</complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
TTL	Time To Live	No		
comment	Ignored			
data	The RData portion of the record to delete. This *must* match the RData exactly.	Yes	-91	Data field required
deviceRecFlag	Ignored			
domain	The name of the domain for this resource record	Yes	-60 -135	Domain not found: *domain* Domain required: *domain*
domainType	The domain type of the domain. Defaults to "Default".	No	-81	Domain type not found: *domainType*
id	Ignored			
Owner	The owner field of the record to be deleted. This must match exactly.	Yes	-89 -134	Owner field required Invalid character in Owner *owner*
resourceRecClass	The Resource Record class. This defaults to "IN".	No		
resourceRecType	The type of resource record being deleted	Yes	-90 -92	Type field required Unknown type: *type*

Other returnCodes and faultstrings

ReturnCode	Faultstring
-2	Exception reading domain or resource record (database error)
-124	DNS Resource record not found
-125	DNS Resource record not unique
401	<401>Unauthorized

DeleteNetElement

Overview

The **DeleteNetElement** API enables the web service client to delete network element from IPAM.

Request and Response Messages

The portion of *Deletes.wsdl* that describes the **deleteNetElement** request and response messages is shown following.

```
<wsdl:operation name="deleteNetElement" parameterOrder="inpNE">
  <wsdl:input message="impl:deleteNetElementRequest" name="deleteNetElementRequest"/>
  <wsdl:output message="impl:deleteNetElementResponse" name="deleteNetElementResponse"/>
</wsdl:operation>
<wsdl:message name="deleteNetElementRequest">
  <wsdl:part name="inpNE" type="tns1:WSNetElement"/>
</wsdl:message>
</wsdl:message>
  <wsdl:message name="deleteNetElementResponse">
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSNetElement** , which is passed as input from the client to the web service, is described in the next section. Only the first two elements are used for this API.

Parameters

WSetElement

The portion of D*eletes.wsdl* that describes **WSNetElement**, the parameter structure passed to **DeleteNetElement**, is shown following. The elements are described in the table that follows.

```
<complexType name="WSNetElement">
  <sequence>
    <element name="agentName" nillable="true" type="soapenc:string"/>
    <element name="authPassword" nillable="true" type="soapenc:string"/>
    <element name="globalSync" nillable="true" type="soapenc:string"/>
    <element name="interfaceNameList" nillable="true" type="impl:ArrayOf_soapenc_string"/>
    <element name="ipAddress" nillable="true" type="soapenc:string"/>
    <element name="model" nillable="true" type="soapenc:string"/>
    <element name="name" nillable="true" type="soapenc:string"/>
    <element name="readCommunityString" nillable="true" type="soapenc:string"/>
    <element name="telnetPassword" nillable="true" type="soapenc:string"/>
    <element name="telnetUser" nillable="true" type="soapenc:string"/>
    <element name="threshold" nillable="true" type="soapenc:string"/>
    <element name="type" nillable="true" type="soapenc:string"/>
    <element name="v3AuthPassword" nillable="true" type="soapenc:string"/>
    <element name="v3AuthProtocol" nillable="true" type="soapenc:string"/>
    <element name="v3ContextName" nillable="true" type="soapenc:string"/>
    <element name="v3EngineId" nillable="true" type="soapenc:string"/>
    <element name="v3PrivacyPassword" nillable="true" type="soapenc:string"/>
    <element name="v3PrivacyProtocol" nillable="true" type="soapenc:string"/>
    <element name="vendor" nillable="true" type="soapenc:string"/>
```

```
  </sequence>
</complexType
```

Element	Accepted Values	Required	Return Code	Faultstring
name	The name of the Network Element. This can be any combination of letters and numbers.	Yes, unless a unique IP address is specified	-33 -35	Network element not found by name: *name* Network Element Name is required *(indicates null input)*
ipAddress	The IP address or fully-qualified domain name (FQDN) of the Network Element. This must be a valid IPv4 or IPv6 IP address, or a full-qualified host name.	Yes, unless the name is specified	-33 -28	Network element not found by ip address: *addr* Specify network element name, not unique by ip address: *addr*

Other returnCodes and faultstrings

ReturnCode	Faultstring
-2	Exception reading network element record (database error)
401	<401>Unauthorized

DeleteNetElementInterface

Overview

The **DeleteNetElementInterface** API enables the web service client to delete network element interfaces from IPAM.

Request and Response Messages

The portion of *Deletes.wsdl* that describes the **deleteNetElementInterface** request and response messages is shown following.

```
<wsdl:operation name="deleteNetElementInterface" parameterOrder="inpNEI">
  <wsdl:input message="impl:deleteNetElementInterfaceRequest"
    name="deleteNetelementInterfaceRequest"/>
  <wsdl:output message="impl:deleteNetElementInterfaceResponse"
    name="deleteNetelementInterfaceResponse"/>
</wsdl:operation>
<wsdl:message name="deleteNetElementInterfaceRequest">
  <wsdl:part name="inpDev" type="tns2:WSNetElementInterface"/>
</wsdl:message>
<wsdl:message name="deleteNetElementInterfaceResponse">
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSNetElementInterface** , which is passed as input from the client to the web service, is described below. For consistency, this is the same structure that is passed to **ImportNetElementInterface**. However, only two of the fields are used.

Parameters

WSNetElementInterface

The portion of *Deletes.wsdl* that describes **WSNetElementInterface**, the parameter structure passed to **DeleteNetElementInterface**, is shown following. The elements are described in the table that follows.

```
<complexType name="WSNetElementInterface">
  <sequence>
    <element name="id" nillable="true" type="soapenc:int"/>
    <element name="interfaceName" nillable="true" type="soapenc:string"/>
    <element name="netElementName" nillable="true" type="soapenc:string"/>
    <element name="status" nillable="true" type="soapenc:string"/>
  </sequence>
</complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
id	The internal identifier for this network element interface object. If this is not set, a new interface is created. If this is set, the interface with the matching identifier is updated.	No		
interfaceName	The name of the interface being added or modified.	Yes	-39 -20	Cannot delete interface: *interface* for network element: *name* because there are blocks attached. Network Element Interface Name is required
netElementName	The name of a Network Element already defined to IPAM.	Yes	-33 -35	Network element not found: *netelement* Network Element name is required
status	The status of the interface. This can be one of "Disabled", "Enabled", or "Deployed". The default on an import is "Enabled".	No	-34	Invalid interface status: *status*

DeleteNetService

Overview

The **DeleteNetService** API enables the web service client to delete network service from IPAM.

Request and Response Messages

The portion of *Deletes.wsdl* that describes the **deleteNetService** request and response messages is shown following.

```
<wsdl:operation name="deleteNetService" parameterOrder="inpNS">
  <wsdl:input message="impl:deleteNetServiceRequest"
    name="deleteNetServiceRequest"/>
  <wsdl:output message="impl:deleteNetServiceResponse"
    name="deleteNetServiceResponse"/>
</wsdl:operation>
<wsdl:message name="deleteNetServiceRequest">
  <wsdl:part name="inpNS" type="tns1:WSNetService"/>
</wsdl:message>
<wsdl:message name="deleteNetServiceResponse">
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSNetService**, which is passed as input from the client to the web service, is described below. Only the first two elements are used for this API.

Parameters

WSNetService

The portion of *Deletes.wsdl* that describes **WSNetService**, the parameter structure passed to **DeleteNetService**, is shown following. The elements are described in the table that follows.

```
<complexType name="WSNetService">
  <sequence>
    <element name="agentName" nillable="true" type="soapenc:string" />
    <element name="collectionMethod" nillable="true" type="soapenc:string" />
    <element name="collectionPort" nillable="true" type="soapenc:string" />
    <element name="container" nillable="true" type="impl:ArrayOf_soapenc_string" />
    <element name="globalSync" nillable="true" type="soapenc:string" />
    <element name="ipAddress" nillable="true" type="soapenc:string" />
    <element name="name" nillable="true" type="soapenc:string" />
    <element name="threshold" nillable="true" type="soapenc:string" />
    <element name="type" nillable="true" type="soapenc:string" />
    <element name="userName" nillable="true" type="soapenc:string" />
    <element name="userPassword" nillable="true" type="soapenc:string" />
    <element name="vendor" nillable="true" type="soapenc:string" />
    <element name="vendorInfo" nillable="true" type="impl:ArrayOf_soapenc_string"/>
  </sequence>
</complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
name	The name of the Network Service	Yes	-185 -186	Network Service not found for name:*name* with type: *type* Network Service Name is required *(indicates null input)*
type	The type of Network Service. If this column is left blank, **dhcp** is assumed.	No	-187	Invalid network service type: *type*

Other returnCodes and faultstrings

ReturnCode	Faultstring
-2	Exception reading network service record (database error)
401	<401>Unauthorized

DeleteNetworkElement

Overview

The **deleteNetworkElement** API enables the web service client to delete a network element from IPAM. By specifying the name of the network element to be deleted, the web service will identify and delete the network element.

Request and Response Messages

The portion of *Deletes.wsdl* that describes the **deleteNetworkElement** request and response messages is shown following.

```
<wsdl:message name=" deleteNetworkElementResponse">
</wsdl:message>
<wsdl:message name=" deleteNetworkElementRequest">
      <wsdl:part name="inpNE" type="tns2:WSNetworkElement"/>
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSNetworkElement**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSNetworkElement

The portion of *Deletes.wsdl* that describes **WSNetworkElement**, the parameter structure passed to **deleteNetworkElement**, is shown following. The required elements are described in the table that follows. The other elements are ignored.

```
<complexType name="WSNetworkElement">
    <sequence>
     <element name="SNMPRetries" nillable="true" type="soapenc:int"/>
     <element name="SNMPTimeout" nillable="true" type="soapenc:int"/>
     <element name="SNMPVersion" nillable="true" type="soapenc:string"/>
     <element name="SNMPsysDescr" nillable="true" type="soapenc:string"/>
     <element name="SNMPsysLocation" nillable="true" type="soapenc:string"/>
     <element name="SNMPsysName" nillable="true" type="soapenc:string"/>
     <element name="SNMPsysServices" nillable="true" type="soapenc:string"/>
     <element name="agentNames" nillable="true" type="impl:ArrayOf_soapenc_string"/>
     <element name="description" nillable="true" type="soapenc:string"/>
     <element name="globalSync" type="xsd:boolean"/>
     <element name="id" nillable="true" type="soapenc:int"/>
     <element name="interfaceNames" nillable="true" type="impl:ArrayOf_soapenc_string"/>
     <element name="interfaceStatus" nillable="true" type="impl:ArrayOf_soapenc_string"/>
     <element name="interfaceTemplate" nillable="true" type="soapenc:string"/>
     <element name="ipAddress" nillable="true" type="soapenc:string"/>
     <element name="model" nillable="true" type="soapenc:string"/>
     <element name="name" nillable="true" type="soapenc:string"/>
     <element name="readCommunityString" nillable="true" type="soapenc:string"/>
     <element name="type" nillable="true" type="soapenc:string"/>
     <element name="v3AuthPassword" nillable="true" type="soapenc:string"/>
     <element name="v3AuthProtocol" nillable="true" type="soapenc:string"/>
     <element name="v3ContextName" nillable="true" type="soapenc:string"/>
```

```
      <element name="v3EngineId" nillable="true" type="soapenc:string"/>
      <element name="v3PrivacyPassword" nillable="true" type="soapenc:string"/>
      <element name="v3PrivacyProtocol" nillable="true" type="soapenc:string"/>
      <element name="v3Username" nillable="true" type="soapenc:string"/>
      <element name="vendor" nillable="true" type="soapenc:string"/>
    </sequence>
</complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
name	The name of the network element to be deleted.	Yes, unless a unique IP address is specified.	-33	Network Element not found. input name: *name* input ipAddress: *address*
			-35	Network Element Name is required
			-211	Network element is associated with a device container. Delete the device container instead.
			-212	Unable to delete Network element: input name: *name* / input ipAddress: *address* Network element has blocks
ipAddress	The ipAddress of the network element to be deleted.	Yes, unless the name is specified.	-28	Specify network element name, not unique by ip address: *address*

Other returnCodes and faultstrings

ReturnCode	Faultstring
-2	Exception deleting network element record (database error)
-318	Admin is not authorized to access Network Element functions

DeleteNetworkLink

Overview

The **deleteNetworkLink** API enables the web service client to delete a network link from IPAM By specifying the name of the link to be deleted, the web service will identify and delete the network link.

Request and Response Messages

The portion of *Deletes.wsdl* that describes the **deleteNetworkLink** request and response messages is shown following.

```
<wsdl:message name="deleteNetworkLinkResponse">
</wsdl:message>
<wsdl:message name="deleteNetworkLinkRequest">
      <wsdl:part name="inpNetLink" type="tns2:WSNetworkLink"/>
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSNetworkLink**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSNetworkLink

The portion of *Deletes.wsdl* that describes **WSNetworkLink**, the parameter structure passed to **deleteNetworkLink**, is shown following. The required elements are described in the table that follows. The other elements are ignored.

```
<complexType name="WSNetworkLink">
    <sequence>
     <element name="blockNames" nillable="true" type="impl:ArrayOf_soapenc_string"/>
     <element name="containers" nillable="true" type="impl:ArrayOf_soapenc_string"/>
     <element name="description" nillable="true" type="soapenc:string"/>
     <element name="dhcpOptionSet" nillable="true" type="soapenc:string"/>
     <element name="dhcpPolicySet" nillable="true" type="soapenc:string"/>
     <element name="id" nillable="true" type="soapenc:string"/>
     <element name="name" nillable="true" type="soapenc:string"/>
     <element name="type" nillable="true" type="soapenc:string"/>
    </sequence>
   </complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
Name	The name of the network link to be deleted.	Yes	-287 -279 -291	Network Link Name is required Network Link not found for name: *name* Admin is not

DeleteNetworkLink

Element	Accepted Values	Required	Return Code	Faultstring
				authorized to access Network Link functions

DeletePrefixPool

Overview

The **DeletePrefixPool** API enables the web service client to delete Prefix Pools from IPAM. This will also delete the Delegated Prefixes inside the Prefix Pool.

Request and Response Messages

The portion of *Deletes.wsdl* that describes the **DeletePrefixPool** request and response messages is shown following.

```
<wsdl:operation name="deletePrefixPool" parameterOrder="inpPrefixPool">
 <wsdl:input message="impl:deletePrefixPoolRequest" name="deletePrefixPoolRequest"/>
 <wsdl:output message="impl:deletePrefixPoolResponse"  name="deletePrefixPoolResponse"/>
</wsdl:operation>

<wsdl:message name="deletePrefixPoolRequest">
  <wsdl:part name="inpPrefixPool" type="tns2:WSPrefixPool"/>
</wsdl:message>

 <wsdl:message name="deletePrefixPoolResponse">
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex types **WSPrefixPool**, which are passed as input from the client to the web service, are described in the next section. For consistency, these are the same structures that are passed to **ImportPrefixPool**. However, only two of the fields are used.

Parameters

WSPrefixPool

The portion of *Deletes.wsdl* that describes **WSPrefixPool**, the parameter structure passed to **DeletePrefixPool**, is shown following. The elements are described in the table that follows.

```
<complexType name="WSPrefixPool">
 <sequence>
  <element name="allowClientClasses" nillable="true" type="impl:ArrayOf_soapenc_string"/>
  <element name="container" nillable="true" type="soapenc:string"/>
  <element name="delegatedPrefixLength" nillable="true" type="soapenc:int"/>
  <element name="denyClientClasses" nillable="true" type="impl:ArrayOf_soapenc_string"/>
  <element name="dhcpOptionSet" nillable="true" type="soapenc:string"/>
  <element name="dhcpPolicySet" nillable="true" type="soapenc:string"/>
  <element name="id" nillable="true" type="soapenc:int"/>
  <element name="length" nillable="true" type="soapenc:int"/>
  <element name="longestPrefixLength" nillable="true" type="soapenc:int"/>
  <element name="name" nillable="true" type="soapenc:string"/>
  <element name="primaryNetService" nillable="true" type="soapenc:string"/>
  <element name="shortestPrefixLength" nillable="true" type="soapenc:int"/>
  <element name="startAddr" nillable="true" type="soapenc:string"/>
  <element name="type" nillable="true" type="soapenc:string"/>
 </sequence>
</complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
Name	String	Yes	-262	Prefix Pool Not found
Start Address	String	Yes	-262	Prefix Pool Not found

Other returnCodes and faultstrings

ReturnCode	Faultstring
-2	Exception processing delete (database error)
-99	Access Denied

DeleteTaskByDate

Overview

The **DeleteTaskByDate** API enables the web service client to delete tasks from IPAM that are older than a given date.

Request and Response Messages

The portion of *Deletes.wsdl* that describes the **DeleteTaskByDate** request and response messages is shown following.

```
<wsdl:operation name="deleteTaskByDays" parameterOrder="before">
  <wsdl:input message="impl:deleteTaskByDateRequest" name="deleteTaskByDateRequest"/>
  <wsdl:output message="impl:deleteTaskByDateResponse" name="deleteTaskByDateResponse"/>
</wsdl:operation>

<wsdl:message name="deleteTaskByDateRequest">
  <wsdl:part name="before" type="xsd:dateTime"/>
</wsdl:message>

<wsdl:message name="deleteTaskByDateResponse">
  <wsdl:part name="deleteTaskByDateReturn" type="xsd:int"/>
</wsdl:message>
```

Response

The count of tasks deleted is returned.

Request

The request takes one parameter which is a date. Any tasks older than this date are deleted.

Parameters

Element	Accepted Values	Required	Return Code	Faultstring
Before	Date/Time	Yes	-107 -110	Missing Date Invalid Date

DeleteTaskByDays

Overview

The **DeleteTaskByDays** API enables the web service client to delete tasks from IPAM that are older than a given number of days.

Request and Response Messages

The portion of *Deletes.wsdl* that describes the **DeleteTaskByDays** request and response messages is shown following.

```
<wsdl:operation name="deleteTaskByDays" parameterOrder="days">
  <wsdl:input message="impl:deleteTaskByDaysRequest" name="deleteTaskByDaysRequest"/>
  <wsdl:output message="impl:deleteTaskByDaysResponse" name="deleteTaskByDaysResponse"/>
</wsdl:operation>

<wsdl:message name="deleteTaskByDaysRequest">
  <wsdl:part name="days" type="xsd:int"/>
</wsdl:message>

<wsdl:message name="deleteTaskByDaysResponse">
  <wsdl:part name="deleteTaskByDaysReturn" type="xsd:int"/>
</wsdl:message>
```

Response

The count of tasks deleted is returned.

Request

The request takes one parameter which is the number of days of tasks to retain. Any tasks older than the current date minus this parameter are deleted.

Parameters

Element	Accepted Values	Required	Return Code	Faultstring
Days	A positive integer.	Yes	-109	Invalid Days value

DeleteTaskById

Overview

The **DeleteTaskByID** API enables the web service client to delete one or more tasks from IPAM.

Request and Response Messages

The portion of *Deletes.wsdl* that describes the **DeleteTaskById** request and response messages is shown following.

```
<wsdl:operation name="deleteTaskById" parameterOrder="ids">
  <wsdl:input message="impl:deleteTaskByIdRequest" name="deleteTaskByIdRequest"/>
  <wsdl:output message="impl:deleteTaskByIdResponse" name="deleteTaskByIdResponse"/>
</wsdl:operation>

<wsdl:message name="deleteTaskByIdRequest">
  <wsdl:part name="ids" type="impl:ArrayOf_soapenc_int"/>
</wsdl:message>

<wsdl:message name="deleteTaskByIdResponse">
  <wsdl:part name="deleteTaskByIdReturn" type="xsd:int"/>
</wsdl:message>
```

Response

The count of tasks deleted is returned.

Request

The request takes one parameter, which is an array of integers. Each integer is a Task ID.

Parameters

Element	Accepted Values	Required	Return Code	Faultstring
Ids	The Task IDs to delete, one per element in the array.	Yes	-106 -108	Task Not found Missing IDs

DeleteZone

Overview

The **DeleteZone** API enables the web service client to delete zones from IPAM.

Request and Response Messages

The portion of *Deletes.wsdl* that describes the **deleteZone** request and response messages is shown following.

```
<wsdl:message name="deleteZoneRequest">
  <wsdl:part name="inpZone" type="tns2:WSDnsZone"/>
</wsdl:message>
<wsdl:message name="deleteZoneResponse">
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSDnsZone**, which is passed as input from the client to the web service, is described in the next section.

Parameters

WSDnsZone

The portion of *Deletes.wsdl* that describes **WSDnsZone**, the parameter structure passed to **DeleteZone**, is shown following. The elements are described in the table that follows.

```
<complexType name="WSDnsZone">
  <sequence>
    <element name="MNAME" nillable="true" type="soapenc:string"/>
    <element name="acceptZoneTransfers" nillable="true" type="soapenc:string"/>
    <element name="aliasZone" type="xsd:boolean"/>
    <element name="allowUpdateACL" nillable="true" type="soapenc:string"/>
    <element name="autogenNSGlue" nillable="true" type="soapenc:string"/>
    <element name="domainName" nillable="true" type="soapenc:string"/>
    <element name="domainType" nillable="true" type="soapenc:string"/>
    <element name="dynamicZone" type="xsd:boolean"/>
    <element name="filename" nillable="true" type="soapenc:string"/>
    <element name="galaxyName" nillable="true" type="soapenc:string"/>
    <element name="masters" nillable="true" type="soapenc:string"/>
    <element name="publishNS" nillable="true" type="soapenc:string"/>
    <element name="server" nillable="true" type="soapenc:string"/>
    <element name="templateZone" type="xsd:boolean"/>
    <element name="updatePolicy" nillable="true" type="soapenc:string"/>
    <element name="updateZone" nillable="true" type="soapenc:string"/>
    <element name="view" nillable="true" type="soapenc:string"/>
    <element name="zoneExtensionsAfter" nillable="true" type="soapenc:string"/>
    <element name="zoneExtensionsPrior" nillable="true" type="soapenc:string"/>
    <element name="zoneType" nillable="true" type="soapenc:string"/>
  </sequence>
</complexType>
```

Element	Description and accepted values	Required	Return Code	Faultstring
MNAME	Ignored	No		
acceptZoneTransfers	Ignored	No		
aliasZone	Ignored	No		
allowUpdateACL	Ignored	No		
autogenNSGlue	Ignored	No		
domainName	Domain name for the zone.	Yes	-135 -60	Domain required Domain not found: *domainType/domain*
domainType	Domain type of the domain for the zone. If not specified, the "Default" domain type will be used.	Yes, when not "Default".	-81	Domain type not found: *domainType*
dynamicZone	Ignored	No		
Filename	Ignored	No		
galaxyName	Ignored	No		
Masters	Ignored	No		
publishNS	Ignored	No		
Server	The network service name of the DNS Server for the zone.	Yes	-235 -93	Server name required Server not found: *server*
templateZone	Ignored	No		
updatePolicy	Ignored	No		
updateZone	Ignored	No		
View	The name of the view in which the zone exists. If not specified, 'Default' will be used.	Yes, when not "Default".	-58	View not found: *view*
zoneExtensionsAfter	Ignored	No		
zoneExtensionsPrior	Ignored	No		
zoneType	Ignored	No		

Other returnCodes and faultstrings

ReturnCode	Faultstring
-2	Various - Exception reading domain or domain type record (database error)
-88	Zone not found: *domainName* / *domainType* / *server* / *view*
-99	Access denied

DeleteZoneResourceRecord

Overview

The **DeleteZoneResourceRecord** API enables the web service client to delete resource records from IPAM that are affiliated with a zone. These are specialized resource records, known as "glue" records.

Request and Response Messages

The portion of *Deletes.wsdl* that describes the **deleteZoneResourceRecord** request and response messages is shown following.

```
<wsdl:operation name="deleteZoneResourceRecord" parameterOrder="inpRR">
  <wsdl:input message="impl:deleteZoneResourceRecordRequest"
    name="deleteZoneResourceRecordRequest"/>
  <wsdl:output message="impl:deleteZoneResourceRecordResponse"
    name="deleteZoneResourceRecordResponse"/>
</wsdl:operation>
<wsdl:message name="deleteZoneResourceRecordRequest">
  <wsdl:part name="inpRR" type="tns2:WSZoneResourceRec"/>
</wsdl:message>
<wsdl:message name="deleteZoneResourceRecordResponse">
</wsdl:message>
```

Response

There is no data in the response.

Request

The complex type **WSZoneResourceRec**, which is passed as input from the client to the web service, is described below.

Parameters

WSResourceRecord

The portion of *Deletes.wsdl* that describes **WSZoneResourceRec**, the parameter structure passed to **DeleteZoneResourceRecord**, is shown following. The elements are described in the table that follows.

```
<complexType name="WSZoneResourceRec">
  <sequence>
    <element name="TTL" nillable="true" type="soapenc:string"/>
    <element name="data" nillable="true" type="soapenc:string"/>
    <element name="owner" nillable="true" type="soapenc:string"/>
    <element name="resourceRecClass" nillable="true" type="soapenc:string"/>
    <element name="resourceRecType" nillable="true" type="soapenc:string"/>
    <element name="server" nillable="true" type="soapenc:string"/>
    <element name="view" nillable="true" type="soapenc:string"/>
    <element name="zone" nillable="true" type="soapenc:string"/>
  </sequence>
</complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
TTL	Time To Live	No		
Data	The RData portion of the record to delete. This *must* match the RData exactly.	Yes	-91	Data field required
Owner	The owner field of the record to be deleted. This must match exactly.	Yes	-89	Owner field required
resourceRecClass	The Resource Record class. This defaults to "IN".	No		
resourceRecType	"A" or "NS"	Yes	-90 -92	Type field missing Type field invalid
server	The DNS Server that serves the zone	Yes		
view	The DNS View in which the zone resides. Defaults to "Default".	No		
zone	The Name of the zone in which the resource record exists.	Yes	-88	Zone not found.

DetachContainer

Overview

The **DetachContainer** API enables the web service client to detach a device container from one of its parents. The device container must have more than one parent. If the device container has one parent, the DeleteContainer API must be used to delete it from the container hierarchy instead.

Request and Response Messages

The portion of *Deletes.wsdl* that describes the **deleteContainer** request and response messages is shown following.

```
   <wsdl:operation name="detachContainer">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="detachContainerRequest">
         <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://diamondip.com/ipcontrol/ws/" use="encoded"/>
      </wsdl:input>
      <wsdl:output name="detachContainerResponse">
         <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://diamondip.com/ipcontrol/ws/" use="encoded"/>
      </wsdl:output>
   </wsdl:operation>
    <wsdl:message name="detachContainerRequest">
      <wsdl:part name="container" type="soapenc:string"/>
      <wsdl:part name="v4DhcpServer" type="soapenc:string"/>
      <wsdl:part name="v6DhcpServer" type="soapenc:string"/>
   </wsdl:message>
   <wsdl:message name="detachContainerResponse">
   </wsdl:message>
```

Response

There is no data in the response.

Request

The request contains three strings to describe the detach operation requested.

Parameters

container

The full path name of the device container to be detached. The full path is required here as that determines which parent to detach the container from. The container must have more than one parent.

v4DhcpServer

The V4 DHCP server to use for any pools or subnets whose DHCP server has been invalidated by the change in parent hierarchy. If this is not specified, no changes will be made to the assigned DHCP servers.

v6DhcpServer

The V6 DHCP server to use for any pools or subnets whose DHCP server has been invalidated by the change in parent hierarchy. If this is not specified, no changes will be made to the assigned DHCP servers.

Other Interfaces

Callout Manager

The Callout Manager is a facility within IPAM that notifies other applications of alerts and programmatic events. Examples of Callout Manager uses are:

- Interfacing to other Alert Management facilities
- Automating Router or DHCP configuration
- Performing off-line auditing

Operation

The Callout Manager performs the following functions:

- Receive a message (via JMS) from the other IPAM components
- Inspects the message to determine the event type
- Takes the data supplied with the event and writes it to a temporary file. The file is written as XML, or as name-value pairs, as dictated by the configuration (see below).
- Execute the script that is configured for this event, passing it the name of the temporary file.

When the message queue is empty, the callout manager simply waits.

Note that the callout manager does not wait for the user script to complete. Hence, the user script must delete the temporary file passed to it.

When building your scripts, if you do not specify fully qualify output paths, then your defined output would be written to `C:\Program Files\Cisco Prime Network Registrar IPAM\etc` on Windows, or `/opt/incontrol/etc` on Linux. Assuming that path is where you installed InControl. For example, if your script had `[echo 'hello world' > output.txt]`, rather than `[echo 'hello world' > /opt/incontrol/tmp/out.txt]`, then the *output.txt* file would be found under `/opt/incontrol/etc`.

Configuration

The Callout Manager is configured through a text file known as a "properties" file. The Callout Manager's properties file can be found in *$INCHOME/callout_manager.properties*. The properties file contains directives that control the Callout Manager behavior. Any

changes made to this file require that the Callout Manager service be restarted for the changes to take effect.

The following table lists those properties. Locate the property or properties within *callout_manager.properties* that you want to use and uncomment it (remove the # in front of the line), and specify a custom path and script name. For example:

```
block.add = /opt/scripts/blockadd_callout.sh
```

Property	Required	Description
log.config.filename	Yes	Specifies the name of the file that holds the logging directives.
queue.connections.names	Yes	Specifies the JMS Queue Name. This should not be changed from its shipped value.
queue.connections.callout.factory.name	Yes	Specifies the Java class that manages the Queue creation. Should not be changed from its shipped value
queue.connections.callout.thread.count	Yes	Specifies the number of threads for receiving callout messages. Should not be changed from its shipped value.
queue.connections.callout.reconnect.retry	Yes	Specifies the reconnection retry count if the Callout manager is disconnected from JMS. Should not be changed from its shipped value.
queue.connections.callout.reconnect.delay	Yes	Specifies the retry interval should the Callout Manager be disconnected from JMS. Should not be changed from its shipped value.
output.path	No	Specifies the directory where the temporary files will be created for the events. Defaults to $INCHOME/tmp. If specified, the string must end with a path separator.
output.xml	No	Specifies the format of the temporary file. If false (default), the file contains name=value pairs. If true, the file contains XML.
alertcallout	No	The name of the command to execute when an alert is raised. For example, on Windows, an example of the property value would be c:\\Program Files\\Cisco Prime Network Registrar IPAM\\bin\\alertcallout.bat. On Linux, an example of the property value would be /opt/incontrol/bin/alertcallout.sh.
block.add	No	The name of the command to execute when a block is added to IPAM. For example, on Windows, an example of the property value would be c:\\Program Files\\Cisco Prime Network Registrar IPAM\\bin\\ blockadd_callout.bat. On Linux, an example of the property value would be /opt/incontrol/bin/blockadd_callout.sh.

Property	Required	Description
block.modify	No	The name of the command to execute when a block is modified within IPAM. For example, on Windows, an example of the property value would be c:\\Program Files\\Cisco Prime Network Registrar IPAM\\bin\\ blockmodify_callout.bat. On Linux, an example of the property value would be /opt/incontrol/bin/blockmodify_callout.sh.
block.delete	No	The name of the command to execute when a block is deleted within IPAM. For example, on Windows, an example of the property value would be c:\\Program Files\\Cisco Prime Network Registrar IPAM\\bin\\ blockdelete_callout.bat. On Linux, an example of the property value would be /opt/incontrol/bin/blockdelete_callout.sh
device.add	No	The name of the command to execute when a device is added within the IPAM GUI. For example, on Windows, an example of the property value would be c:\\Program Files\\Cisco Prime Network Registrar IPAM\\bin\\ deviceadd_callout.bat. On Linux, an example of the property value would be /opt/incontrol/bin/deviceadd_callout.sh
device.modify	No	The name of the command to execute when a device is modified within the IPAM GUI. For example, on Windows, an example of the property value would be c:\\Program Files\\Cisco Prime Network Registrar IPAM\\bin\\ devicemodify_callout.bat. On Linux, an example of the property value would be /opt/incontrol/bin/devicemodify_callout.sh
device.delete	No	The name of the command to execute when a device is deleted within the IPAM GUI. For example, on Windows, an example of the property value would be c:\\Program Files\\Cisco Prime Network Registrar IPAM\\bin\\ devicedelete_callout.bat. On Linux, an example of the property value would be /opt/incontrol/bin/devicedelete_callout.sh
domain.add	No	The name of the command to execute when a domain is added within the IPAM GUI. For example, on Windows, an example of the property value would be c:\\Program Files\\Cisco Prime Network Registrar IPAM\\bin\\ domainadd_callout.bat. On Linux, an example of the property value would be /opt/incontrol/bin/domainadd_callout.sh

Property	Required	Description
domain.modify	No	The name of the command to execute when a domain is modified within the IPAM GUI. For example, on Windows, an example of the property value would be c:\\Program Files\\Cisco Prime Network Registrar IPAM\\bin\\ domainmodify_callout.bat. On Linux, an example of the property value would be /opt/incontrol/bin/domainmodify_callout.sh
domain.delete	No	The name of the command to execute when a domain is deleted within the IPAM GUI. For example, on Windows, an example of the property value would be c:\\Program Files\\Cisco Prime Network Registrar IPAM\\bin\\ domaindelete_callout.bat. On Linux, an example of the property value would be /opt/incontrol/bin/domaindelete_callout.sh
workflow.rr.create.readyforreview	No	The name of the command to execute when an admin creates a resource record that needs approval. For example, on Windows, an example of the property value would be c:\\Program Files\\Cisco Prime Network Registrar IPAM\\bin\\wf_rr_create_callout.bat. On Linux, an example of the property value would be /opt/incontrol/bin/wf_rr_create_callout.sh
workflow.rr.update.readyforreview	No	The name of the command to execute when an admin updates a resource record that needs approval. For example, on Windows, an example of the property value would be c:\\Program Files\\Cisco Prime Network Registrar IPAM\\bin\\wf_rr_update_callout.bat. On Linux, an example of the property value would be /opt/incontrol/bin/wf_rr_update_callout.sh
workflow.rr.delete.readyforreview	No	The name of the command to execute when an admin deletes a resource record that needs approval. For example, on Windows, an example of the property value would be c:\\Program Files\\Cisco Prime Network Registrar IPAM\\bin\\wf_rr_delete_callout.bat. On Linux, an example of the property value would be /opt/incontrol/bin/wf_rr_delete_callout.sh

Property	Required	Description
workflow.rr.approved	No	The name of the command to execute when an admin approves a pending resource record. For example, on Windows, an example of the property value would be c:\\Program Files\\Cisco Prime Network Registrar IPAM\\bin\\wf_rr_approved_callout.bat. On Linux, an example of the property value would be /opt/incontrol/bin/wf_rr_approved_callout.sh **Note**: The command will not be executed in the scenario where an approving admin updates or deletes a resource record that is pending approval, since it is **not** considered a 'workflow approval' operation
workflow.rr.rejected	No	The name of the command to execute when an admin rejects a resource record that needs approval. For example, on Windows, an example of the property value would be c:\\Program Files\\Cisco Prime Network Registrar IPAM\\bin\\wf_rr_rejected_callout.bat. On Linux, an example of the property value would be /opt/incontrol/bin/wf_rr_rejected_callout.sh **Note**: The command will not be executed in the scenario where an approving admin updates or deletes a resource record that is pending approval, since it is **not** considered a 'workflow rejection' operation.
workflow.device.create.readyforreview	No	The name of the command to execute when an admin creates a device that needs approval. For example, on Windows, an example of the property value would be c:\\Program Files\\Cisco Prime Network Registrar IPAM\\bin\\wf_device_create_callout.bat. On Linux, an example of the property value would be /opt/incontrol/bin/wf_device_create_callout.sh
workflow.device.delete.readyforreview	No	The name of the command to execute when an admin deletes a device that needs approval. For example, on Windows, an example of the property value would be c:\\Program Files\\Cisco Prime Network Registrar IPAM\\bin\\wf_device_delete_callout.bat. On Linux, an example of the property value would be /opt/incontrol/bin/wf_device_delete_callout.sh

Property	Required	Description
Workflow.device.approved	No	The name of the command to execute when an admin approves a device that needs approval. For example, on Windows, an example of the property value would be c:\\Program Files\\Cisco Prime Network Registrar IPAM\\bin\\wf_device_approved_callout.bat. On Linux, an example of the property value would be /opt/incontrol/bin/wf_device_approved_callout.sh **Note**: The command will not be executed in the scenario where an approving admin deletes a device that is pending approval, since it is **not** considered a 'workflow approval' operation.
Workflow.device.rejected	No	The name of the command to execute when an admin rejects a device that needs approval. For example, on Windows, an example of the property value would be c:\\Program Files\\Cisco Prime Network Registrar IPAM\\bin\\wf_device_rejected_callout.bat. On Linux, an example of the property value would be /opt/incontrol/bin/wf_device_rejected_callout.sh **Note**: The command will not be executed in the scenario where an approving admin deletes a device that is pending approval, since it is **not** considered a 'workflow rejection' operation.
task.complete	No	The name of the command to execute when a task is launched with the IPAM GUI. For example, on Windows, an example of the property value would be c:\\Program Files\\Cisco Prime Network Registrar IPAM\\bin\\taskcomplete_out.bat. On Linux, an example of the property value would be /opt/incontrol/bin/taskcomplete_callout.sh

RIR Template Support

Introduction

IPAM includes support for creating a limited set of Regional Internet Registry (RIR) templates that can be electronically mailed to the appropriate registry. This includes templates for ARIN and RIPE. This support is provided via sample scripts that can be called via the IPAM Callout Manager. The scripts include the ability to automatically send an email with the appropriate content to the RIR.

For further details about RIR templates, visit the ARIN and/or RIPE websites at http://www.arin.net or http://www.ripe.net.

Configuration

ARIN File Generation

IPAM provides a set of sample scripts that can be used, via the Callout Manager, to generate the proper ARIN Reassign - Simple template, commonly referred to as a SWIP (Shared WhoIs Project) template. The Callout Manager can be configured to call these scripts on each Add, Delete, or Modify of a block within IPAM. To configure the callout manager to call the SWIP scripts on these events, modify the following properties in the *$INCHOME/callout_manager.properties* file. (The examples assume that *$INCHOME = /opt/incontrol*.)

```
block.add = /opt/incontrol/etc/callout/addSwip.pl
block.modify = /opt/incontrol/etc/callout/modifySwip.pl
block.delete = /opt/incontrol/etc/callout/deleteSwip.pl
```

The scripts can be configured with default information to be used when generating the data files. The default properties are stored in a properties file called *$INCHOME/etc/callout/swip.properties*. The following table lists those properties.

Property	Required	Description
from_address	Yes	The email address of the sender of the template email.
to_address	Yes	The email address of the recipient of the email address. Typically, this should be reassign@arin.net.
subject	No	The subject of the email. For a Reassign – Simple template, this should be "REASSIGN SIMPLE"
customer_name	No	Specifies the default name of the customer to which the block has been reassigned.
customer_address	No	Specifies the default street address for the customer being assigned this block.
customer_city	No	Specifies the default city for the customer being assigned this block.
customer_state	No	Specifies the default state for the customer being assigned this block.
customer_postal_code	No	Specifies the default postal code for the customer being assigned this block.
customer_country_code	No	Specifies the default country code for the customer being assigned this block.

RIPE File Generation

IPAM provides a set of sample scripts that can be used, via the Callout Manager, to generate the proper RIPE inetnum or inet6num templates. These templates are used to update the RIPE database directly via email. The Callout Manager can be configured to call these scripts on each Add, Delete, or Modify of a block within IPAM. To configure the callout manager to call the RIPE scripts on these events, modify the following properties in the *$INCHOME/callout_manager.properties* file. (The examples assume that *$INCHOME = /opt/nc*.)

```
block.add = /opt/nc/etc/callout/addRipe.pl
block.modify = /opt/nc/etc/callout/modifyRipe.pl
block.delete = /opt/nc/etc/callout/deleteRipe.pl
```

The scripts can be configured with default information to be used when generating the data files. The default properties are stored in a properties file called *$INCHOME/etc/callout/ripe.properties*. The following table lists those properties.

Property	Required	Description
from_address	Yes	The email address of the sender of the template email.
to_address	Yes	The email address of the recipient of the email address. Typically, this should be reassign@arin.net.
orgname	No	Specifies the text that will appear in the "descr" field of the in inetnum (or inet6num) templates. Typically this is the name of the organization that will use the address space.
adminc	No	Specifies the email address of the administrative contact for this address space. This corresponds to the "admin-c" field in the inetnum (or inet6num) template.
techc	No	Specifies the email address of the technical contact for this address space. This corresponds to the "tech-c" field in the inetnum (or inet6num) template.
revsrv	No	Specifies the reverse server address for this address space. This corresponds to the "rev-srv" field in the inetnum (or inet6num) template.
notify	No	Specifies the email address of the RIPE contact for this address space. This corresponds to the "notify" field in the inetnum (or inet6num) template.
status	No	Specifies the default values of the status field in the inetnum or inet6num templates. All **inetnum** objects under APNIC Whois Database must have a status attribute. The status attribute must be one of the following values: UNSPECIFIED ALLOCATED PORTABLE ALLOCATED NON-PORTABLE ASSIGNED PORTABLE ASSIGNED NON-PORTABLE All **inet6num** objects under APNIC Whois Database must have a status attribute. The status attribute must be one of the following values: ALLOCATED PORTABLE ALLOCATED NON-PORTABLE ASSIGNED PORTABLE ASSIGNED NON-PORTABLE

Property	Required	Description
mntby		The default value to use as the identifier of a registered mntner object used for authorization and authentication.
mntlower		Sometimes there is a hierarchy of maintainers. In these cases, mnt-lower is used as well as mnt-by. This field specifies the default value to use.
mntroutes		The identifier of a registered mntner object used to control the creation of route objects associated with the address range specified by the inetnum (or inet6num) object.
mntirt		The name of an irt object that represents a Computer Security Incident Response Team (CSIRT) that handles security incidents for the address space specified by the inetnum object.
remarks		General remarks. May include a URL or instructions on where to send abuse complaints.
changed		The email address of who last updated the database object and the date it occurred. The changed attribute is not a network contact address, as it merely records who made the most recent change to the registration information. All RIPE addresses will initially record an RIPE address in this attribute, as RIPE creates the first database object.
source		The name of the database from which the data was obtained. This is either "RIPE" or a designation for the NIR, LIR or ISP that allocates/assigns the address.
password		The default password to use if the either the CRYPT-PW or MD5 authentication is used with the mntner object.
delete_reason		The default text to use when deleting an address space. This will be placed in the "Delete:" field, if a delete has taken place.

Operation

ARIN File Generation Scripts

In order to support the automatic file generation of ARIN SWIP templates and emails, there are three scripts that can get called by the Callout Manager when a block is added, modified, or deleted and they are addSwip.pl, modifySwip.pl, and deleteSwip.pl. (For information on configuring the Callout Manager to perform these tasks please see the section above called Configuration – ARIN File Generation.)

All of the scripts are written in PERL and require that PERL 5 be installed and configured on the IPAM Executive system by the system administrator. PERL 5 is NOT supplied with IPAM. Although fully functional, the scripts are provided as examples.

The ARIN scripts all operate in a similar fashion. They accept as their first argument a filename. This filename should point to a file that contains block information formatted as name-value pairs. The scripts parse this file to pull out any required data and decide whether or not to proceed with the template processing. The key decision point is the presence of a property called **`swipname`** in the data file. If the property is present and is non-empty, then the scripts will proceed with creating a Reassign-Simple template for adding, modifying, or deleting an address space. After creating the template, it will attempt

to send an email to the address specified in the **`to_address`** field, as defined in the *swip.properties* file.

Note: All scripts delete the data file before exiting.

Expected User Defined Field Mappings

The SWIP Perl scripts expect to make use of both standard and User Defined Fields. The following table shows a mapping of the fields used by the scripts and their relation to the standard and User Defined Fields.

UDF / IPAM Tags	SWIP Field	UDF
startaddrstring / blocksize	IP Address and Prefix	N
swipname	Network-Name	N
org_name	Customer Name	Y
street_address	Customer Address	Y
city	Customer City	Y
state	Customer State	Y
postal_code	Customer Postal Code	Y
country_code	Customer Country Code	Y
public_comments	Public Comments	Y

RIPE File Generation Scripts

In order to support the automatic file generation of RIPE inetnum or inet6num templates and emails, there are three scripts that can get called by the Callout Manager when a block is added, modified, or deleted and they are addRipe.pl, modifyRipe.pl, and deleteRipe.pl. (For information on configuring the Callout Manager to perform these tasks please see the section above called Configuration – RIPE File Generation.)

All of the scripts are written in PERL and require that PERL 5 be installed and configured on the IPAM Executive system by the system administrator. PERL 5 is NOT supplied with IPAM. Although fully functional, the scripts are provided as examples.

The RIPE scripts all operate in a similar fashion. They accept as their first argument a filename. This filename should point to a file that contains block information formatted as name-value pairs. The scripts parse this file to pull out any required data and decide whether or not to proceed with the template processing. The key decision point is the presence of a property called "**`swipname`**" in the data file. If the property is present and is non-empty, then the scripts will proceed with creating an inetnum or inet6num template for adding, modifying, or deleting an address space. After creating the template, it will attempt to send an email to the address specified in the **`toaddress`** field, as defined in the *ripe.properties* file.

Note: All scripts delete the data file before exiting.

Expected User Defined Field Mappings

The RIPE Perl scripts expect to make use of both standard and User Defined Fields. The following table shows a mapping of the fields used by the scripts and their relation to the standard and User Defined Fields.

UDF / IPAM Tags	RIPE Field	UDF
startaddrstring - endaddrstring	inetnum (or inet6num)	N
swipname	netname	N
orgname	descr	Y
country_code	country	Y
admin_contact	admin-c	Y
tech_contact	tech-c	Y
rev_srv	rev-srv	Y
Status	status	Y
remarks	remarks	Y
Notify	notify	Y
Mntby	mnt-by	Y
mntlower	mnt-lower	Y
mntroutes	mnt-routes	Y
changed	changed	Y
Source	Source	Y

RIR REST Interface Support

Introduction

IPAM includes support for creating a limited set of Regional Internet Registry (RIR) templates that can be electronically mailed to the appropriate registry. Please see the previous section called RIR Template Support for full details. Since those scripts were created, ARIN and RIPE have implemented REST APIs for updating their databases. IPAM has implemented a set of example scripts that can be run via the Callout Manager to make use of these APIs.

For further details about RIR REST Interfaces, visit the ARIN and/or RIPE websites at https://www.arin.net/resources/whoisrws/whois_api.html or https://www.ripe.net/support/documentation/developer-documentation.

Description

When a child block is added, modified or deleted in IPAM, the Callout Manager executes a CLI for each operation if it is configured to do so. The CLIs update the block information to the appropriate RIR registry using a REST API (if supported by the RIR). A Single argument is passed to the CLIs, which is the name of a file that holds all the information about the IPv4/IPv6 address block. The example scripts provided will make use of the REST APIs provided by either ARIN or RIPE.

Configuration

Before using the callout scripts, there are some steps that must be performed to configure the system to perform the callouts and to use information specific to your organization. For a few of these steps, database SQL scripts have been provided to automate the creation of User Defined Fields (UDFs) and Information Templates. The instructions below all assume the default location of INCHOME as /opt/incontrol. Adjust as necessary to suit your environment.

Modify callout_manager.properties and restart Callout Manager.

Edit /opt/incontrol/callout_manager.properties file and add the following lines at the end of the file. Also make sure the properties are not defined anywhere in the file.

```
##################################################
block.add = /opt/incontrol/etc/callout/rir/addBlock.sh
block.modify = /opt/incontrol/etc/callout/rir/modifyBlock.sh
block.delete = /opt/incontrol/etc/callout/rir/deleteBlock.sh
##################################################
```

These scripts are wrapper scripts around the single program rirCallout.rb which is written using the Ruby language. It is executed using the Java JVM installed with IPAM using JRuby, also included.

Once the Callout Manager is configured, it needs to be restarted.

To restart Callout Manager, type:

```
>/opt/incontrol/etc/cm restart
```

Create the required User Defined Fields (UDFs) in IPAM.

The RIR scripts make use of information native to IPAM Blocks and RIR Organizations as well as User Defined Fields (UDFs) that contain data needed by the internet registries. SQL scripts are provided for both MySQL and Oracle that will create the UDFs along with the Information Templates that group the UDFs. The information templates are discussed in the next section. The scripts are located in the `$INCHOME/etc/callout/rir` directory and are named RirUdfs.sql and ora_RirUdfs.sql for MySQL and Oracle respectively.

For MySQL, it is executed by running the '`mysql`' utility located in $INCHOME/mysql/bin:

```
>./mysql –uincadmin –pxxxx incontrol < $INCHOME/etc/callout/rir/RirUdfs.sql
```

For Oracle, it is executed by running the '`sqlplus`' utility. (The location will vary per organization.)

```
>sqlplus incadmin/xxxx @$INCHOME/etc/callout/rir/ora_RirUdfs.sql
```

The table below details each of the UDFs.

Display Title	Field Name/Tag	Value Data Type	RIR	Template	Description
Allocated	allocated	Checkbox	ARIN	Arin Block Info	If the Checkbox is selected, that means that the block will be Reallocated to an organization. The default is NOT selected and indicates a Reassignment to a customer.
Org Handle	orghandle	Text	ARIN	Arin Block Info	This field is used as orgHandle in XML payload. If the Allocated Checkbox is selected, a valid Org Handle

Display Title	Field Name/Tag	Value Data Type	RIR	Template	Description
					must be specified in this field.
NetHandle	nethandle	Text	ARIN	Arin Block Info	This field is used to save the netHandle after adding the Block to the ARIN database. The netHandle is used to delete the Block.
Customer Handle	customerhandle	Text	ARIN	Arin Block Info	This field is used to save the customerHandle after a customer is created and a block is successfully assigned to the customer. The customerHandle is used to delete the customer from the ARIN database if the address block is deleted.
Comment	comment	Text	ARIN, RIPE	Arin Block Info, Ripe Block Info	The comment used in the block is saved after the block is saved to the RIR database. If the block could not be saved to RIR database, the error message is saved in this field.
Organization	org_orgname	Text	RIPE	Ripe	This UDF will

Display Title	Field Name/Tag	Value Data Type	RIR	Template	Description
ID				Block Info	be used as the "descr" attribute for RIPE.
RIR Status	rirstatus	Textarea	ARIN, RIPE	Arin Block Info, Ripe Block Info	This UDF is used to record the status of the REST operation to the RIR database. In case of an error, the error message is stored in this UDF.
Parent Net Handle	nethandle	Text	ARIN	Root Block Info	This UDF is necessary for ARIN. The UDF must be populated on the root block with the net handle of the root block as it is in ARIN database. For example NET-10-100-0-1, NET6-2600-1 etc. The value of the UDF is used as parentNetHandle while creating a customer and assigning a block to ARIN.
Maintainer	mntby	Text	RIPE	Root Block Info	This UDF is necessary for RIPE and must be populated on

Display Title	Field Name/Tag	Value Data Type	RIR	Template	Description
					the Root block. It is used in the "mnt-by" attribute.
Customer Name	customer_name	Text	ARIN	Arin Block Info	This UDF represents the customer name and a mandatory field for Reassigned blocks, ie. Not Allocated.
Customer Address 1	customer_address1	Text	ARIN	Arin Block Info	This UDF represents the first line of address of a customer and a mandatory field for Reassigned blocks, ie. Not Allocated.
Customer Address 2	customer_address2	Text	ARIN	Arin Block Info	This UDF represents the second line of address of a customer and a optional field.
Customer City	customer_city	Text	ARIN	Arin Block Info	This UDF represents the city of a customer and a mandatory field for Reassigned blocks, ie. Not Allocated.
Customer State/Province	customer_state_province	Text	ARIN	Arin Block Info	This UDF represents the state of a

Display Title	Field Name/Tag	Value Data Type	RIR	Template	Description
					customer and a mandatory field for Reassigned blocks, ie. Not Allocated.
Customer Postal Code	customer_postal_code	Text	ARIN	Arin Block Info	This UDF represents the postal code of a customer and a mandatory field for Reassigned blocks, ie. Not Allocated.
Customer Country Code	customer_country_code	Text	ARIN	Arin Block Info	This UDF represents the two letter country (e.g. US) customer and a mandatory field for Reassigned blocks, ie. Not Allocated.
Customer Is Private	customer_is_private	Checkbox	ARIN	Arin Block Info	This UDF specifies if the cutomer data is private or not. The default value is unchecked, that is the customer data is public.

Create Information Templates

Create three Information Templates, one for each of the RIRs ARIN and RIPE, one for root blocks. The appropriate UDFs from the previous step will be assigned to each of the templates. The SQL scripts mentioned above will create the templates and assign the UDFs to them. The UDF assignments are detailed below.

Arin Block Info

Allocated

OrgHandle

Net Handle

Customer Handle

Comment

RIR Status

Customer Name

Customer Address1

Customer Address2

Customer City

Customer State/Pronice

Customer Postal Code

Customer Country Code

Customer is Private

Ripe Block Info

Comment

RIR Status

Organization ID

Root Block Info

Parent Net Handle

Maintainer

Fill up required fields in the RIR Organization

RIR Organization can be created or edited by clicking on Tools->RIR Organization IDs menu item. The following table details the fields that must be filled out properly.

Field Name	ARIN	RIPE	Required	Description
Organization ID	Yes	No	Yes	The value is used as "orgHandle" for a Reassigned ARIN address block. It is a required field. Also this value

Field Name	ARIN	RIPE	Required	Description
				must be a registered value in ARIN database.
Admin Contact	No	Yes	Yes	This value is used as "admin-c" for a RIPE address block. It is a required field.
Tech Contact	No	Yes	Yes	This value is used as "tech-c" for a RIPE address block. It is a required field.
Password	No	Yes	Yes	This value is used as authorization information for RIPE. It is a required field.
Email Update To	No	Yes	No	If specified, the value will be used as the changed attribute for RIPE. If it is not specified, the value is used from INI file.
Notify Email To	No	Yes	No	If specified, the value will be used as the "notify" attribute for RIPE. RIPE usually sends email to this address if

Field Name	ARIN	RIPE	Required	Description
				any object is added or modified successfully.

Configure Root Blocks with Organizations and Internet Registries

If the specific block information will be updated to an RIR database, make sure that the root block of the block has a RIR Organization associated with it. In addition you must make sure that the Internet Registry field is set to the appropriate registry (ARIN or RIPE).

A root block must have an RIR organization associated with it, if the allocated or assigned child blocks are to managed in an RIR database. For ARIN, Organization ID (for orgHandle) is used from RIR Organization. For RIPE, Admin Contact, Tech Contact and Password are used from RIR Organization. For RIPE, Email update to will be used for the "changed" attribute if specified and Notify Email to will be used for "notify" if specified.

Similarly, to enable the RIR Callout scripts to communicate with the correct registry, the Internet Registry field must be set to either ARIN or RIPE, for all Root blocks obtained from these registries.

To configure a Root Block with an RIR Organization:

1) Navigate to the container where the root block resides using the Management -> Container View menu option.
2) Click the "notepad with pencil" icon next to the appropriate block.
3) Select the Internet Registry from the list box.
4) Select the Organization ID from the list box.
5) Click the "Save" button.

Update the INI File.

There is an initialization file called `rir_callout.ini` located by default in `/opt/incontrol/etc/callout/rir`. The INI file looks like a regular Windows INI file. Any line that starts with a ';' or a '#' is considered a comment. The properties are specified as key = value pair. The pairs are used to provide information ranging from parameters required to communicate with the RIR APIs to default values for some of the fields. The file has 3 sections [COMMON], [RIPE] and [ARIN]. Any key = value pair in [COMMON] can be overwritten in the [RIPE] or [ARIN] sections.

Some important key = value pairs that must be set specifically for your environment include:

[COMMON]

```
;;-------------------------------------------------------------------------
; Database user used by the above CLIs
```

```
db_user = incadmin

;;-------------------------------------------------------------------------
; Database user's password used by the above CLIs
db_password = incadmin
;;-------------------------------------------------------------------------
; Error email address/es. If REST operation fails, send email to the
; address/es if specified. If more than one email address needs to
; be specified, separate the addresses by a comma or a semicolon or
; add more than one error_email_to lines. Note: if notify email is
; specified in RIR Org, it will be used as well.
; Example:
; error_email_to = jdoe@example.com, mjane@example.com; foo@example.com
error_email_to = jdoe@example.com, mjane@example.com; foo@example.com

[RIPE]
;;-------------------------------------------------------------------------
; For test database it is "TEST", for Production change it to "RIPE"
; This property must exist.
source = TEST
;;-------------------------------------------------------------------------
; RIPE's required changed attribute.
;;changed = noc@example.com
changed =  noc@example.com

;;-------------------------------------------------------------------------
; If Maintainer is not defined as UDF in rootblock. Use this value
; as last resort.
;maintainer = TEST123-MNT
maintainer = TEST123-MAINT-RIPE

;;-------------------------------------------------------------------------
; country code. Hardcode to US
; This propery must exist.
country = US

;;-------------------------------------------------------------------------
; RIPE API's can take multiple passwords
; password for maintainer
; pasword is also taken from RIR org table
password = Test4567
;;-------------------------------------------------------------------------
; password for lower maintainer. For test database it is the
; password of TEST-DBM-MNT
password = emptypassword

[ARIN]
;;-------------------------------------------------------------------------
; ARIN RESt API Key.
```

```
api_key = XXXXXXXXXXXXXXXXXXXXXX
```

Please look at APPENDIX B for a sample INI file. Each of the parameters are fully explained in the file. Please browse through it and change the policies as needed to suit your environment.

Enable SWIP/Net Name for the container and block type combination

There is a policy in the rir_callout.ini file called ignore_if_no_swip_netname which defaults to true. The implication of this policy is that the RIR Callout scripts will only attempt to modify the RIR database if the value of the SWIP / Netname field is not empty. In order to enable the entry of the SWIP/Netname value for blocks in specific containers, you must configure the selected Container and Block Type combinations. To do this:

1) Choose the Management -> Container Maintenance menu option.

2) Navigate to the desired container.

3) Click the Edit Container link.

4) Click the Require SWIP / Net Name tab

5) Select the checkbox for each Block Type for which you want to require SWIP / Net Name entry.

6) Click the "Submit" button to save your changes.

Associate Information Template with the container and blocktype combination.

In order to enable the entry of the RIR User Defined Field values for blocks in specific containers, you must assign the appropriate Information Templates to the selected Container and Block Type combinations. To do this:

1) Choose the Management -> Container Maintenance menu option.

2) Navigate to the desired container.

3) Click the Edit Container link.

4) Click the Block Type Information Templates tab.

5) For each Block Type for which you want to assign the UDFs, choose the appropriate Information Template from the list box.

6) Once you have configured all of the desired Block Types, click the "Submit" button to save your changes.

Notes about Block Modification

It is important to know what can be modified in a block. Different fields can be modified for ARIN and RIPE.

ARIN Block Modification

A sample Net payload for ARIN is shown below:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<net xmlns="http://www.arin.net/regrws/core/v1">
  <pocLinks/>
  <customerHandle>C0123456</customerHandle>
  <comment>
    <line number="0">rirCallout v1.03, Tue Sep 11 16:08:34 -0400 2012</line>
  </comment>
  <netBlocks>
    <netBlock>
      <cidrLength>24</cidrLength>
      <description>Reassigned</description>
      <endAddress>123.094.254.255</endAddress>
      <startAddress>123.094.254.000</startAddress>
      <type>S</type>
    </netBlock>
  </netBlocks>
  <handle>NET-123-94-254-0-1</handle>
  <netName>SAMPLE-BSN-BEQUICK-123-94-254-0-Y</netName>
  <originASes/>
  <parentNetHandle>NET-123-94-0-0-1</parentNetHandle>
  <registrationDate>2012-09-11T16:08:35-04:00</registrationDate>
  <version>4</version>
 </net>
```

As per ARIN Net Payload documentation, the following objects can **not** be modified:

- `version`
- `customerHandle`
- `orgHandle`
- `netBlock`
- `parentNetHandle`

Note: `customerHandle` is used for simple reassignment of an address block to a customer and `orgHandle` is used for reallocation to an organization and they are mutually exclusive.

Also the "`handle`" (Net Handle) can not be modified. That means for ARIN, only the following fields can be modified:

- `netName` (SWIP/Net Name)

- `comment`

Before sending the modification payload to ARIN, first the existing NET payload is retrieved and compared for change. If anything has changed, the retrieved "`registrationDate`" attribute is added to the payload and sumbitted via REST.

Note: Never modify `Net Handle` or `Customer Handle` in IPAM GUI as they are used for deleting the block and the customer from ARIN database. When a block is modified in IPAM GUI, the change is saved in the database first, therefore, the rirCallout program can not undo the change. You may consider changing these these UDFs to hidden in a production environment.

RIPE Block Modifications

Only "netname" (SWIP/Net Name) can be modified for RIPE. Although RIPE accepts comment in the "remarks" attribute, it never shows up in the RIPE database, so we do not send it.

DNS Listener

The DNS Listener process can dynamically collect information from DNS servers about updates to zones and import this information into IPAM.

The DNS Listener process imports DNS resource records collected via an incremental zone transfer (IXFR) from a DNS server into IPAM. It does this by sequentially reading each resource record contained in an IXFR, processing each one according to a set of rules described below, and then inserting some portion of the resulting data into IPAM. Resource records are processed using rules specific to each resource record's Type field. The type-specific rules are listed in detail further down in this appendix.

Usage

Starting the DNSListener on Unix

```
$INCHOME/etc/dl_start [-c <listener.properties>]
```

Stopping the DNSListener on Unix

```
$INCHOME/etc/dl_stop
```

Starting or Stopping DNSListener via Windows GUI

Follow these steps.

1. Click on **Start -> Control Panel -> Administrative Tools**.
2. Double-click on **Administrative Tools**.
3. Double-click on **Services**.
4. Scroll down to InControl DNS Listener.
5. Right-click on **InControl DNS Listener**.
6. Choose which status you want, either **Start** or **Stop**.

Configuration

Configure the DNS Listener

1. Make sure the listener is not running by using the appropriate stop command (see "Usage" above).
2. The Listener listens on port 5053 by default. If you require it to listen on another port, edit the *dns_listener.properties* file.

Note: For the Listener to listen on a port numbered from 1-1023 on UNIX, you must run the Listener as root so that the process can access privileged ports.

3. Start the DNS Listener by using the appropriate start command (see "Usage" above).

Configure DNS

BIND 8.0 or newer

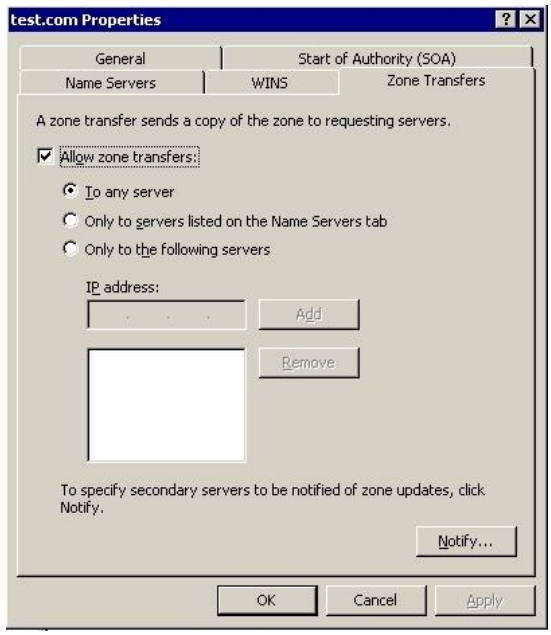
In the BIND configuration file, usually */etc/named.conf*, add the **`also-notify`** option to the zones that you want IPAM to stay synchronized with. For example, the **`also-notify`** option in this file would cause bind to send notify messages to the address 192.168.0.1 on port 5053 when it updates zones **`example.com.`** and **`0.168.192.in-addr.arpa.`**:

```
Options {
                          directory "/var/lib/named";
                          notify no;
};

zone "example.com." {
                          type master;
                          file "db.example";
                          also-notify { 192.168.0.1 port 5053; };
};

zone "0.168.192.in-addr.arpa." {
                          type master;
                          file "db.0.168.192";
                          also-notify { 192.168.0.1 port 5053; };
};
```

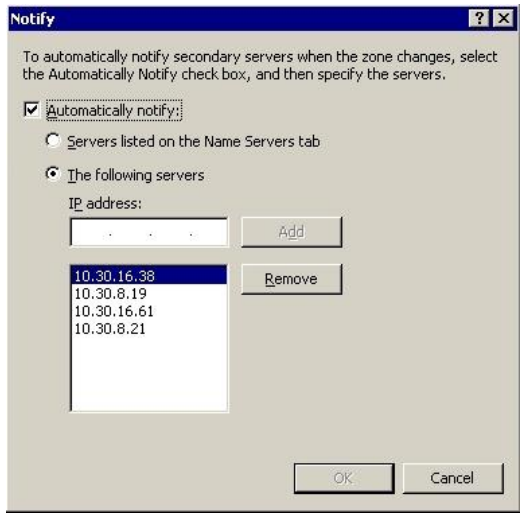
Microsoft DNS Server

Follow these steps.

1. Start the Microsoft DNS Server application: **Control Panel->Administrative Tools->DNS**
2. Click the plus symbol '+', next to the machine icon to open the zones that server is responsible for.
3. Open the Forward Lookup Zones or Reverse Lookup Zones.
4. Right click on the name of the zone that you want to keep synchronized in IPAM and select **Properties** from the menu.



4. Click on the Zone Transfers tab and then click the **Notify...** button.



5. Select **The following servers** from the radio button group.
6. Add the IP address of the DNS Listener, and any other DNS servers that you want notified when zones are updated.
7. Click **OK**.

Note: Microsoft DNS can only send Notify messages on port 53, so the Listener will need to be configured to listen on port 53 by editing the dns_listener.properties file.

Record Processing Rules

RR Type	Description
SOA	Data from the SOA record is used to update a domain in IPAM. The domain name is taken from the name field of the resource record. If the domain does not exist in IPAM processing for that record stops. If the domain exists in IPAM, data from the imported record is used to update the serial number of the domain.
A	Data from A records is used to create resource record entries attached to domains in IPAM and additionally can create individual IP addresses and devices. Note that: • if a domain name that matches the zone name is not defined in IPAM then processing of the resource record stops. • if the name field contains the zone name, with additional fields to the left of the domain name then an IP address and device will be created. • if the name field matches exactly the zone name then no IP address and device will be created. • if an address and device is to be created the DNS Listener will use the left most label of the name field as the host portion of the FQDN for the host. • if the address found in the rdata field of the resource record can be located within a block defined in IPAM, an IP address will be created using the rdata field, and a device will be created using the name field. The device and IP address will also be associated with the resource record in addition to the domain. • if no block is defined that contains the address then the address device pair will not be created. If the IP address already exists in a block defined in IPAM and there is no host name associated with the linked device, the host name will be taken from the name field of the resource record.
PTR	Data from PTR records is used in the same way as data from A records with the exception that the name and rdata field are swapped when creating an IP address and device.
NS	NS records are ignored by the DNS Listener daemon.
MX	MX records are imported into IPAM and attached to the domain supplied in the name field.
SRV	The data from SRV records are processed in the same way as other resource records with the exception of the name field. The name field of SRV records specifies a service available for the domain for which it is a part of. The service type and protocol is encoded in the left portion of its name field. To avoid collision with the rest of the domain name space, leading under-bars are prepended to the labels that describe the service. This practice is not always followed in the field and so the **`DNSImport`** CLI uses the following rule to determine where the domain name part of the name field starts. It considers all the labels to the right of the right-most label that starts with an underscore to be part of the domain name of the SRV record. For example in the following SRV record: `_ldap._tcp.pdc._msdcs.sw.ins.com. 600 SRV 0 100 400 pdc.sw.ins.com.` The service specification part would be: `_ldap._tcp.pdc._msdcs` and the domain name part would be: `sw.ins.com.`
All others	The domain name that the resource record will be placed in is taken from the name field of the resource record after the left label has been removed. If the domain can not be found in IPAM processing of the resource record will stop. If the domain is found in IPAM, a resource record object is created using the data supplied by the imported record, and it is linked to the domain.

Detailed Description

The DNS Listener is a small multi-threaded program that listens for messages from DNS servers. When notified of a change to a zone that a server is responsible for the listener can request detailed information about those changes and then use this information to update IPAM. In this way IPAM is kept synchronized with the DNS as hosts join and leave the network.

The listener is composed of three long lived threads, one queue, and a short lived thread. See Figure 1. The notify thread is responsible for listening for notify messages from DNS servers. When changes are made to the affected zones the DNS server will send a NOTIFY message to the listener. Notify messages sent by DNS servers are sent to port 53 by default, which is a privileged port on most UNIX systems. For this reason the DNS Listener defaults to port 5053. Both the port the server sends to and the port the listener listens on can be changed as described below.

When the listener receives a NOTIFY message, the notify thread sends a message to the transfer manager thread which increments its count of notify messages for the server. When the number of notify messages exceeds the listener.notifythreshold property, and the transfer manager is not in the paused state, the transfer manager thread will create and start a transfer thread. The transfer thread will request an IXFR from the appropriate DNS server, place the resulting resource records on the queue, and then exit.

The resource record input manager thread is notified when records are placed on the queue. When this happens the input manager compares the number of records on the queue with the listener.maxrecords property. If there are more records than the listener.maxrecords specifies, the input manager thread will send a pause message to the transfer manager thread. When the transfer manager is in the pause state it will continue to process notify messages from the notify thread, but will not start any new transfer threads. The input thread will then sequentially remove the resource records from the queue and process them as described above. When the number of records remaining on the queue is less than the listener.maxrecords, a restart message will be sent to the transfer manager.

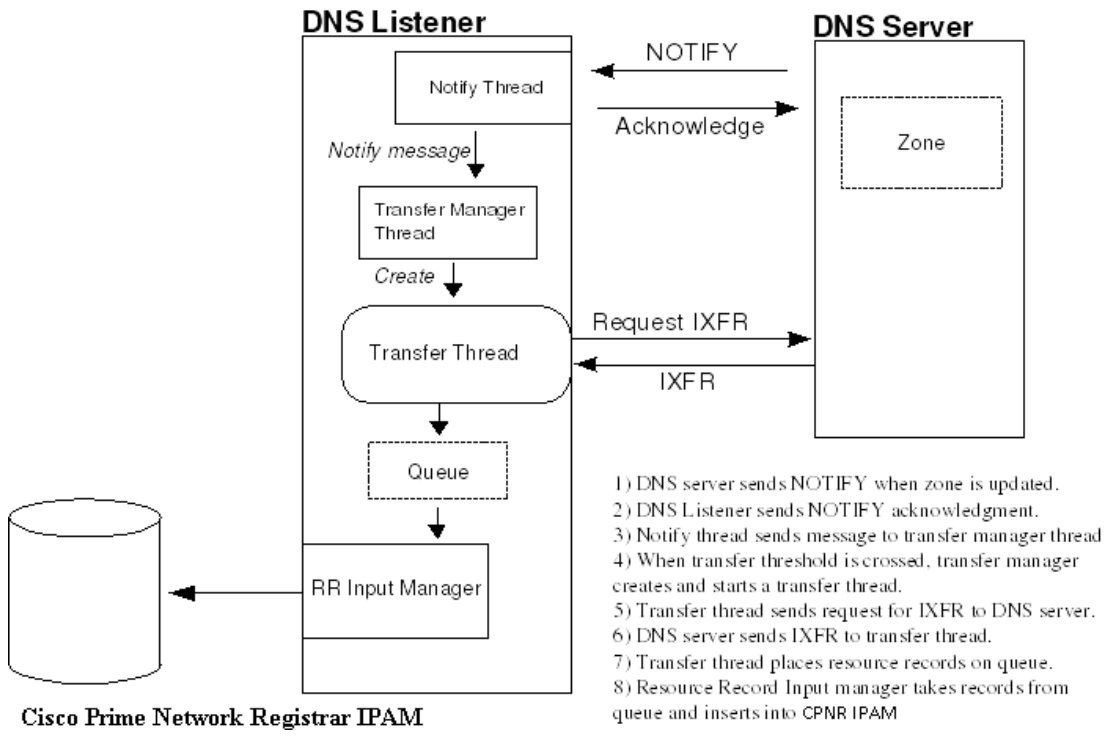


Figure 1 – DNS Listener

The DNS Listener can use a properties file to set certain operational properties. The properties are:

listener.port — The port on which the listener should listen for NOTIFY messages from a DNS server. The default is 5053 if this property is not supplied.

Note: For the listener to listen on port 53 on *NIX, you must run dl_start as root.

listener.notifythreshold — The number of NOTIFY messages received before the listener attempts an IXFR transfer from the DNS server. The default is one NOTIFY message.

listener.maxrecords — High water mark that controls NOTIFY message processing. If more than **`listener.maxrecords`** is accepted by the listener via an IXFR further IXFR requests will not be initiated until all records have been processed by the queue processing thread. This helps limit the amount of memory the listener can claim at any one time. The default is 1000 records.

DNS Deployment Callout

When a DNS File-based deployment task is performed and the configuration and data files are placed on the DNS Server, the IPAM Agent has the ability to execute a callout script. The details of this script are as follows:

- The script is called by the remote agent, that is, the one that resides on the actual DNS server.
- The name of the script is not configurable. It is always *$INCHOME/etc/dns_callout.sh* (or *%INCHOME%\etc\dns_callout.cmd* for Windows).
- The script gets called just before we attempt to Restart the server (on a `DNS` **`Config - All Files push`**) or call **`rndc`** (for Selected/Changed Zone File-based pushes).
- The script gets passed one parameter which is the full path name of the new *named.conf* file.
- The agent will wait for the completion of script before moving on, but only for at most 60 seconds.
- The agent will report the return code of the script in the task result messages, however it does not interrogate this return value and thus will always continue even if the script fails.

Appendix A API Changes

IPAM 8.3

Below is the list of changes to the APIs for IPAM 8.3 that may affect web service clients implemented at your site.

WSInterface

Two elements were added to **WSInterface**, the parameter structure passed to **importDevice,** and returned by **exportDevice** and **getDevice**:

```
<element name="relayAgentCircuitId" nillable="true" type="impl:ArrayOf_soapenc_string"/>
<element name="relayAgentRemoteId" nillable="true" type="impl:ArrayOf_soapenc_string"/>
```

Refer to the I**mportDevice** section of the API chapter for more detailed information.

exportAllNetServicesAsCSV

The name of this API operation was corrected from exportAllNetSevicesAsCSV. (The "r" in Services was missing.)

WSAddrpool

One element was added to **WSAddrpool**, the parameter structure passed to **importAddressPool,** and returned by **getAddressPool**:

```
<element name="overlapInterfaceIp" type="xsd:boolean"/>
```

Refer to the I**mportAddressPool** section of the API chapter for more detailed information.

WSPrefixPool

One element was added to **WSPrefixPool**, the parameter structure passed to **importPrefixPool,** and returned by **getPrefixPool**:

```
<element name="overlapInterfaceIp" type="xsd:boolean"/>
```

Refer to the I**mportPrefixPool** section of the API chapter for more detailed information.

IPAM 8.1.3

Below is the list of changes to the APIs for IPAM 8.1.3 that may affect web service clients implemented at your site.

WSDnsZone

This structure includes a new data member, publishNS, described in **ImportDnsZone** and **DeleteZone** sections of the API chapter.

WSChildBlock

One element, **`nonBroadcast`**, was added to **WSChildBlock**.

```
<element name="nonBroadcast" type="xsd:boolean"/>
```

Refer to the **ImportChildBlock** API section of the API chapter for more detailed information. This structure is used by the **ImportChildBlock, ExportChildBlock** and **DetachBlock**APIs.

WSContainer

One element, **`informationTemplate`**, was modified to be an array:

```
 <element name="informationTemplate" nillable="true"
type="impl:ArrayOf_soapenc_string"/>
```

Additional details are provided in the **ImportContainer** API section. This will affect the **ModifyContainer** CLI as well as the **ImportContainer** and **ExportContainer** APIs.

WSGenericBlock

One element was added to WSGenericBlock:

```
<element name="nonBroadcast" type="xsd:boolean"/>
```

Refer to the **ModifyBlock** API section for more detailed information. This affects the **ModifyBlock** and **GetBlock** APIs.

WSDomainResourceRec

One element was added to WSDomainResourceRec:

```
<element name="deviceRecFlag" type="xsd:boolean"/>
```

Refer to the **ExportDomainResourceRecord** API section for more detailed information. This value is ignored on import.

ImportNetElement

This **ImportNetElement** API and CLI have been deprecated as of IPAM 8.1.3. Use the **ImportNetworkElement** API and CLI instead. ImportNetElement support will be removed in a future release.

ExportNetElement

This **ExportNetElement** API and CLI have been deprecated as of IPAM 8.1.3. Use the **ExportNetworkElement** API and CLI instead. ExportNetElement support will be removed in a future release.

DeleteNetElement

This **DeleteNetElement** API and CLI have been deprecated as of IPAM 8.1.3. Use the **DeleteNetworkElement** API and CLI instead. DeleteNetElement support will be removed in a future release.

DHCPConfigurationAllFiles

This **DHCPConfigurationAllFiles** API and CLI have been updated to add the ability to push only if the configuration has changed. The **DHCPConfigurationAllFiles** CLI and API sections have been updated to reflect the changes.

WSDevice

Note that devices with only the default interface will export the **`container`** field in the **WSInterface** structure. This has been true of the following fields since IPAM 6.0: **`MACAddress`**, **`hwtype`**, **`ipAddress`** and **`excludeFromDiscovery`**. These fields are no longer be exported as part of the **WSDevice** structure. Please refer to the **exportDevice** section of the API chapter for more detailed information.

DetachBlock

The DetachBlock API can now accept values in the interfaceAddress field. If the block has multiple interface addresses on the given Device Container, and an interface address is specified,then the block will be detached from that specified interface address. The block will remain attached to the remaining interfaces addresses.

The DetachBlock CLI now accepts another field in the input file for the interfaceAddress. The DetachBlock CLI and API sections have been updated to reflect the changes.

IPAM 8.1.2

WSAddrpool

One element was added to **WSAddrpool**, the parameter structure passed to **importAddrpool**:

```
<element name="prefixLength" nillable="true" type="soapenc:string"/>
```

In addition, the **sharename** element has been marked for deprecation and will be ignored beginning with IPAM 8.1.2.

Refer to the **importAddrpool** section of the API chapter for more detailed information.

WSAllocationTemplateDetails

The **sharename** element of **WSAllocationTemplateDetails** has been marked for deprecation and will be ignored beginning with IPAM 8.1.2.

This affects the **AddSite** and **ModifyBlock** APIs.

WSAggregateBlock

The **interfaceAddress** element of **WSAggregateBlock** has been marked for deprecation and will be ignored beginning with IPAM 8.1.2.

This affects the **ImportAggregateBlock** API. This field was always ignored by the **DeleteAggregateBlock** API**.**

WSChildBlock

One element, **primarySubnet**, was added to **WSChildBlock**. One element, **interfaceAddress**, was modified to be an array:

```
<element name="interfaceAddress" nillable="true"
   type="impl:ArrayOf_soapenc_string"/>
<element name="primarySubnet" type="xsd:boolean"/>
```

Refer to the **ImportChildBlock** API section of the API chapter for more detailed information. This structure is used by the **ImportChildBlock, ExportChildBlock** and **DetachBlock**APIs.

WSContainer

One additional element, **replaceDHCPServerV6**, was added to **WSContainer**.

```
<element name="replaceDHCPServerV6" nillable="true" type="soapenc:string"/>
```

Additional details are described in the **ImportContainer** API section. This will affect the **ModifyContainer** CLI as well as the **ImportContainer** and **ExportContainer** APIs.

WSDevice

One element was added to **WSDevice**, the parameter structure passed to **importDevice,** and returned by **exportDevice** and **getDevice**:

```
<element name="duid" nillable="true" type="soapenc:string"/>
```

Refer to the **importDevice** section of the API chapter for more detailed information.

Note that devices with only the default interface will export the following fields in the **WSInterface** structure: **MACAddress**, **hwtype**, **ipAddress** and **excludeFromDiscovery**. These fields will no longer be exported as part of the **WSDevice** structure. Please refer to the **exportDevice** section of the API chapter for more detailed information.

WSDhcpServer

One element was added to **WSDhcpServer**, the parameter structure passed to I**mportDhcpServer**:

```
<element name="v4V6Both" nillable="true" type="soapenc:string"/>
```

Refer to the **ImportDhcpServer** section of the API chapter for more detailed information. This affects the **ImportDhcpServer**, **ModifyDhcpServer** and **GetDhcpServer** APIs.

WSDnsZone

Two elements were added to **WSDnsZone**, the parameter structure passed to **importDnsZone**:

```
<element name="dynamicZone" type="xsd:boolean"/>
<element name="updatePolicy" nillable="true" type="soapenc:string"/>
```

Refer to the **importZone** section of the API chapter for more detailed information. This structure is also used by the **deleteZone** API.

WSGenericBlock

One element was added to WSGenericBlock:

```
<element name="primarySubnet" type="xsd:boolean"/>
```

Refer to the **ModifyBlock** API section for more detailed information. This affects the **ModifyBlock** and **GetBlock** APIs.

WSInterface

Two elements were added to **WSInterface**, the parameter structure passed to **importDevice,** and returned by **exportDevice** and **getDevice**:

```
<element name="container" nillable="true type="impl:ArrayOf_soapenc_string"/>
<element name="virtual" nillable="true" impl:ArrayOf_soapenc_boolean/>
```

Refer to the I**mportDevice** section of the API chapter for more detailed information.

WSPrefixPool

This is a new structure described in the **ImportPrefixPool** section of the API chapter.

WSSubnetPolicy

One element was added to **WSSubnetPolicy**:

```
<element name="networkLink" nillable="true" type="soapenc:string"/>
```

This affects the **ImportChildBlock**, **ExportChildBlock,** **GetBlock** and **ModifyBlock** APIs.

ImportNetService

This **ImportNetService** API and CLI have been deprecated as of IPAM 8.1.2. Use the **ImportDhcpServer** API and CLI instead. ImportNetService support will be removed in a future release.

Appendix B – RIR Callout

Sample rir_callout.ini File

Below find a sample INI file for configuring the RIR REST Interface.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; INI file for rirCallout, a program to update IP address block information to
; RIR databases.
;
; The syntax of the file is like a Windows INI file. Do not put single or
; double quotes around any values. The case of Section and keys are
; significant.
;
; The file contains three sections [COMMON], [RIPE], and [ARIN]
; The common properties can be specified under [COMMON] section but can be
; overwritten in each section if necessary.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
[COMMON]

;;-------------------------------------------------------------------------
; ignore the block if no swip/netname is specified. Default is true.
ignore_if_no_swip_netname = true

;;-------------------------------------------------------------------------
; ignore if IPv4 block size is >= n. The value will be used if specified.
;ignore_if_ipv4_block_size_gte = 30

;;-------------------------------------------------------------------------
; ignore if IPv block size is >= n. The value will be used if specified.
;ignore_if_ipv6_block_size_gte =

;;-------------------------------------------------------------------------
; The Cisco Prime Network Registrar IPAM CLI used for exporting a root block. The
parent net handle is
; expected to be in the root block as a User Defined Field. Parent Net Handle
; is required for ARIN.
export_rootblock_cli = /opt/incontrol/etc/cli/ExportRootBlock.sh

;;-------------------------------------------------------------------------
; The Cisco Prime Network Registrar IPAM CLI to export a child block
export_childblock_cli = /opt/incontrol/etc/cli/ExportChildBlock.sh
```

```
;;-------------------------------------------------------------------------
; The Cisco Prime Network Registrar IPAM CLI used for exporting a container.
export_container_cli = /opt/incontrol/etc/cli/ExportContainer.sh


;;-------------------------------------------------------------------------
; The Cisco Prime Network Registrar IPAM CLI used for modifying a block.
modifyblock_cli = /opt/incontrol/etc/cli/ModifyBlock.sh

;;-------------------------------------------------------------------------
; Database user used by the above CLIs
db_user = incadmin

;;-------------------------------------------------------------------------
; Database user's password used by the above CLIs
db_password = incadmin

;;-------------------------------------------------------------------------
; Log file path. It is possible to have a separate log file for
; each of the RIR by specifying a logfile in the specific section.
logfile = /opt/incontrol/log/rir.log

;;-------------------------------------------------------------------------
; How often log file to age. Valid values are daily, weekly, monthly
logfile_aging = daily

;;-------------------------------------------------------------------------
; Sync log file or not. Default is buffered. For tailing, set ync_log to
; true
sync_log = true

;;-------------------------------------------------------------------------
; IP address or FQDN of SMTP server. If FQDN is specified, it must
; be resolvable.
#smtp_server = 172.18.33.22
#-smtp_server = mail.example.com

;;-------------------------------------------------------------------------
; The port SMTP server is listening to.
smtp_port = 25

;;-------------------------------------------------------------------------
; If SMTP server requires authentication, specify the username
; and password. If the server does not support authentication,
; you must comment out the smtp_username and smtp_password.
#-smtp_username = ipcontrol
#-smtp_password = password

;;-------------------------------------------------------------------------
; The email address to use as sender. It will show up in From header.
email_from = ipcontrol@example.com
```

```
;;-------------------------------------------------------------------------
; Send error email or not. Default is true. Can be Overwritten in each section
; This policy should be true, because it is the easiest to know that an
; error has occurred.
send_error_email = true

;;-------------------------------------------------------------------------
; Error email address/es. If REST operation fails, send email to the
; address/es if specified. If more than one email address needs to
; be specified, separate the addresses by a comma or a semicolon or
; add more than one error_email_to lines. Note: if notify email is
; specified in RIR Org, it will be used as well.
; Example:
; error_email_to = jdoe@example.com, mjane@example.com; foo@example.com
; error_email_to = test@example.com
; The emails will be consolidated as:
;   jdoe@example.com, mjane@example.com,foo@example.com,test@example.com
error_email_to = jdoe@example.com, !noc@example.com,

;;-------------------------------------------------------------------------
; If true send email to the addresses specified by success_email_to
; Overwrite in specific section
send_success_email = true

;;-------------------------------------------------------------------------
; Send email to these addresses about successfull REST operation.
; Note if notify email is specified in RIR org, it will be used to
; send success email as well.
;success_email_to = jdoe@example.com
success_email_to = jdoe@example.com

;;-------------------------------------------------------------------------
; Delete block tmp file or not. specify true or false.
; If the value is true, the file will be deleted after the operation.
; The block tmp file is created in /opt/incontrol/tmp directory.
delete_block_tmpfile = false

#########################################################################
#  RIPE Section
#########################################################################
[RIPE]
;;-------------------------------------------------------------------------
; For test database it is "TEST", for Production change it to "RIPE"
; This property must exist.
source = TEST

;;-------------------------------------------------------------------------
; Search url
; This property must exist.
```

```
search_url = http://apps.db.ripe.net/whois/search

;;-------------------------------------------------------------------------
; REST create end point.
; This property must exist.
create_url = https://apps.db.ripe.net/whois/create

;;-------------------------------------------------------------------------
; REST modify end point. It requires source.
; source is TEST or RIPE. <source> will be replaced with the source
; property at run time.
; Note: inetnum and inet6num will be added at the end depending on
; the type of block (IPv4 or IPv6)
modify_url = https://apps.db.ripe.net/whois/update/<source>

;;-------------------------------------------------------------------------
; REST delete end point.
; source is TEST or RIPE. <source> will be replaced with the source
; property at run time.
; Note: inetnum and inet6num will be added at the end depending on
; the type of block (IPv4 or IPv6)
delete_url = https://apps.db.ripe.net/whois/delete/<source>

;;-------------------------------------------------------------------------
; RIPE's required changed attribute.
;;changed = noc@example.com
changed =  noc@example.com

;;-------------------------------------------------------------------------
; If Maintainer is not defined as UDF in rootblock. Use this value
; as last resort.
;maintainer = TEST123-MNT
maintainer = TEST123-MAINT-RIPE

;;-------------------------------------------------------------------------
; country code. Hardcode to US
; This propery must exist.
country = US

;;-------------------------------------------------------------------------
; RIPE API's can take multiple passwords
; password for maintainer
; pasword is also taken from RIR org table
password = Test4567
;;-------------------------------------------------------------------------
; password for lower maintainer. For test database it is the
; password of TEST-DBM-MNT
password = emptypassword

;;-------------------------------------------------------------------------
; If true send email to the addresses specified by success_email_to
; and also to the Notify email in RIR Org
```

```
send_success_email = true

;;-------------------------------------------------------------------------
; Log file path
#logfile = /opt/incontrol/log/ripe.log

;;-------------------------------------------------------------------------
; How often log file to age. Valid values are daily, weekly, monthly
#logfile_aging = daily

;;-------------------------------------------------------------------------
; Delete block tmp file or not. specify true or false.
; If the value is true, the file will be deleted after the
; operation.
;delete_block_tmpfile = true

########################################################################
#  ARIN Section
########################################################################
[ARIN]

;;-------------------------------------------------------------------------
; base REST url
;base_url = https://www.arin.net/rest
base_url = https://rest-beta.arin.net/rest

;;-------------------------------------------------------------------------
; base of the REST net end point
;net_url = https://www.arin.net/rest/net
net_url = https://rest-beta.arin.net/rest/net

;;-------------------------------------------------------------------------
; REST reassign end pont.
; parentnethandle will be replaced at run time.
; HTTP method is PUT.
; Content is NET Payload, returns Ticketed Request Payload
;;-reassign_url = https://www.arin.net/rest/net/<parentnethandle>/reassign
reassign_url = https://rest-beta.arin.net/rest/net/<parentnethandle>/reassign

;;-------------------------------------------------------------------------
; REST reallocate end point.
; parentnethandle will be replaced at run time.
; HTTP method is PUT.
; Content is NET Payload, returns Ticketed Request Payload
;;-reallocate_url = https://www.arin.net/rest/net/<parentnethandle>/reallocate
reallocate_url = https://rest-beta.arin.net/rest/net/<parentnethandle>/reallocate

;;-------------------------------------------------------------------------
; REST modify end point.
; <nethandle> will be replaced at run time.
```

```
; returns NET payload
; HTTP method is PUT
;;-modify_url = https://www.arin.net/rest/net/<nethandle>
modify_url = https://rest-beta.arin.net/rest/net/<nethandle>

;;-------------------------------------------------------------------------
; REST delete end point.
; <nethandle> will be replaced at run time.
; HTTP method is DELETE
;;-delete_url = https://www.arin.net/rest/net/<nethandle>
delete_url = https://rest-beta.arin.net/rest/net/<nethandle>

;;-------------------------------------------------------------------------
; ARIN RESt API Key.
; api_key = XXXXXXXXXXXXXXXXXXXXXX
api_key = API-6184-14CB-4122-36DE

;;-------------------------------------------------------------------------
; Log file path
#logfile = /var/log/arin.log
;;-------------------------------------------------------------------------
; How often log file to age. Valid values are daily, weekly, monthly
#logfile_aging = daily

;;-------------------------------------------------------------------------
; Delete block tmp file or not. specify true or false.
; If the value is true, the file will be deleted after the
; operation.
;delete_block_tmpfile = false
```

Appendix C REST API

Using the API

As of IPAM 8.0, IPAM provides *beta* support that enables access to the IPAM web services API via the REST protocol. The interfaces are grouped into Imports, Gets, Tasks, Exports, Deletes and TaskInvocation services. All of the services documented in this guide as supported by the SOAP interface are also supported by this new REST interface. On-line documentation is provided using Swagger, as described in the next section.

To use the REST API, your application will make HTTP requests and process the responses. The IPAM REST API accepts and produces JSON as its data format. The standard HTTP methods GET, POST and DELETE are supported, depending on the API operation.

In the following examples, *Executive* represents the IP address or the DNS name of the IPAM Executive.

URLs have the following structure:

https://*Executive*:8443/inc-rest/api/v1/*service*/*operation*

where *service* is one of Imports, Gets, Exports, Deletes, TaskInvocation, and *operation* is the specific API. For example:

https://*Executive*:8443/inc-rest/api/v1/Deletes/deleteAdmin

With curl:

```
curl -X DELETE --header "Accept: application/json" --header "Authorization: Basic
aW5jYWRtaW46aW5jYWxxxx4="  -d "{
  \"inpAdmin\": {
        \"loginId\": \"adminOne\"}
}" "https://Executive:8443/inc-rest/api/v1/Deletes/deleteAdmin"
```

Swagger Interface and Tools

- IPAM uses Swagger to document its REST interface. Documentation for Swagger, including information about its tools and resources, can be found at:
  http://swagger.io/

- To access the Swagger user interface for IPAM, execute:
  https://Executive:8443/inc-rest/api/docs/index.html

  In the input box, the following will appear:
  https://*Executive*:8443/inc-rest/api/swagger.json

  The IPAM user name and password are required in order to execute any of the operations. This Swagger interface can be very helpful for developing your API. Documentation is provided for each operation and its parameters. You can execute each operation, testing various input parameters and viewing the responses. For more specific details of each of the parameters than is provided in the Swagger interface, refer to the API reference chapter of this guide.

- To see the complete API JSON specification, see:
  https://*Executive*:8443/inc-rest/api/swagger.json

Authentication

As shown in the example curl, above, IPAM uses HTTP Basic Authentication (BASIC_AUTH) using authentication handlers that are invoked before the API operations are called.