Cisco Prime Network Registrar IPAM 8.1.1 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

Text Part Number: OL-27760-01

Contents

Introduction

About This Guide

This guide outlines command line interfaces (CLIs) into Cisco Prime Network Registrar IP Address Management (IPAM) 8.1.1 and application programming interfaces (APIs) to Cisco Prime Network Registrar IP Address Management (IPAM).

Using CLIs extends the effectiveness of the Cisco Prime Network Registrar IPAM Administrator, allowing him or her flexibility to run Cisco Prime Network Registrar 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 Cisco Prime Network Registrar IPAM product using *cron* or Windows Task Scheduler.

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

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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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). 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*).

Note when creating CSV import files, any lines that begin with the pound (#) character are ignored by the Cisco Prime Network Registrar IPAM CLIs.

Available Command Line Interface Matrix

Object	Import	Modify	Delete	Export
Address Pool	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 RR	X	X	X	
Galaxy Domain	X			
Net Element	X		X	X
Net Element Interface	X	X	X	
Net Service	X		X	X
RR Pending Approval		X		X
RR Pending Approval Status				X
Root Block	X	X	X	X
Zone	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 Cisco Prime Network Registrar IPAM.

Usage

Direct

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

Via command script (Unix)

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

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ImportAddrpool.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the CSV 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.	Yes
C	Address Pool Type	One of "Dynamic DHCP", "Automatic DHCP", "Static", "Reserved".	Yes
D	Name	Address Pool name. Defaults to "Start Address-End Address"	No
E	Share Name	The name used to link address pools together.	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	No
H	Failover Net Service	The name of the failover DHCP server that will serve addresses from this pool	No
I	DHCP Option Set	The name of an Option Set used with this pool.	No
J	DHCP Policy Set	The name of a Policy Set used with this pool.	No
K	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
L	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

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, Cisco Prime Network Registrar 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 <adminId> -p <admin password> -f <import filename> [-r <rejects file>]
[-e <error messages>] [-?]
```

Via command script (Unix)

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

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ImportAggregateBlock.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the CSV 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: Cisco Prime Network Registrar IPAM/Texas/Dallas. Long format eliminates ambiguity in cases where there are duplicate container names.	Yes
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	The specific address, or offset from the beginning, for the interface IP address. 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). An offset of 1 is assumed if none is specified.	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.

Col	Field	Accepted Values	Required
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 Cisco Prime Network Registrar IPAM.

Usage

Direct

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

Via command script (Unix)

```
$INCHOME/etc/cli/ImportChildBlock.sh –u <adminId> -p <admin password>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ImportChildBlock.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-?
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the CSV 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 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
```

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: **Cisco Prime Network Registrar IPAM/Texas/Dallas**. Long format eliminates ambiguity in cases where there are duplicate container names.	Yes

ImportChildBlock

Col	Field	Accepted Values	Required
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.	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 Cisco Prime Network Registrar 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, or offset from the beginning, for the interface IP address. 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).	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

Col	Field	Accepted Values	Required
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.	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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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.cisco.com.|dmz.com./External In this example, hr.cisco.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	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)	No

ImportContainer

Overview

The **`ImportContainer`** CLI allows the user to bulk import containers into Cisco Prime Network Registrar IPAM.

Usage

Direct

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

Via command script (Unix)

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

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ImportContainer.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the CSV 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: **Cisco Prime Network Registrar IPAM/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 '/'.	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	The name of a pre-existing information template to be associated with this container.	No
K	User Defined Fields	Specify the values for the UDFs in the container information template, 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

Col	Field	Accepted Values	Required
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 Cisco Prime Network Registrar IPAM. This is used to bulk load information about existing network devices.

Usage

Direct

```
$INCHOME/jre/bin/java –cp $CLASSPATH com.diamondip.ipcontrol.cli.ImportDevice
–u <adminId> -p <admin password> -f <import_file> [-e <error messages>]
[-r <rejects file>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportDevice.sh –u <adminId> -p <admin password> -f <import_file>
[-e <error messages>] [-r <rejects file>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ImportDevice.cmd –u <adminId> -p <admin password> –f <import_file> [-e <error messages>]
[-r <rejects file>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the CSV 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	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 ("|").	Yes
B	Address Type	The address type of this device. Accepted values are: Static, Dynamic DHCP, Automatic DHCP, Manual DHCP, and Reserved. 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.	Yes
C	Host Name	Valid host name or **APPLYNAMINGPOLICY**	Yes
D	Device Type	The name of a device type configured in Cisco Prime Network Registrar IPAM.	Yes, if Hostname specifies use of naming policy

Col	Field	Accepted Values	Required
E	Hardware Type	Specify **Ethernet** or **Token Ring**. When Hardware Type is specified, MAC Address must also be specified.	Yes, for Manual DHCP or if MAC Address is specified
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 for each IP Address in column A.	Yes, if Hardware Type is specified
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 Cisco Prime Network Registrar 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.	Yes, if overlapping space is in use and the block name is ambiguous.
J	Domain Type	Domain type name already defined to Cisco Prime Network Registrar 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.	Yes, for UDFs defined as required fields.

Col	Field	Accepted Values	Required
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
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**. 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.

Example

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

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

ImportDeviceResourceRecord

Overview

The **`ImportDeviceResourceRecord`** CLI allows the user to bulk import DNS resource records for a device into Cisco Prime Network Registrar IPAM.

Usage

Direct

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

Via command script (Unix)

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

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ImportDeviceResourceRecord.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the CSV 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

Col	Field	Accepted Values	Required
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 Cisco Prime Network Registrar IPAM.

Usage

Direct

```
$INCHOME/jre/bin/java –cp $CLASSPATH com.diamondip.ipcontrol.cli.ImportDhcpServer
–u <adminId> -p <admin password> -f <import_file> [-e <error messages>]
[-r <rejects file>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportDhcpServer.sh –u <adminId> -p <admin password> -f <import_file>
[-e <error messages>] [-r <rejects file>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ImportDhcpServer.cmd –u <adminId> -p <admin password> –f <import_file>
[-e <error messages>] [-r <rejects file>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the CSV 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.	Yes
C	Product	The product name of the DHCP Server. This must be one of the products defined in Cisco Prime Network Registrar IPAM.	Yes
D	Agent	The Cisco Prime Network Registrar 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

Col	Field	Accepted Values	Required
G	Configuration Path	The path to the server's configuration file. Must be a legal, fully qualified path name for the host system.	No
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 TRUE to enable dynamic DNS updates when this server issues a lease. Defaults to FALSE.	No
U	DHCP Option Set	The name of an option set defined in Cisco Prime Network Registrar IPAM.	No
V	DHCP Policy Set	The name of a policy set defined in Cisco Prime Network Registrar IPAM.	No
W	DHCP Client Classes	The names of client classes defined in Cisco Prime Network Registrar 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.	

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 <file name> -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 <file name> -s <server> [-v <view>] [-z <zone>] [-l]
[-t <domainType>] [-m <view/zone=domainType, … >] [-n] [-2 None|ZoneOnly|ZoneAndRR]
[-c container]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ImportDNS.cmd -f <file name> -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 <file name>	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 Cisco Prime Network Registrar 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.cisco.com=internal,external/dmz.cisco.com=external"`
		An example when views are not defined:
		`-m "hr.cisco.com=internal,dmz.cisco.com=external"`
		Note: Quotes are not necessary unless there are spaces in the DomainType names. They are used here for completeness.

-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	N	Container name. The *full pathname* (e.g. "Cisco Prime Network Registrar IPAM/North America/US/PA") of a container in Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar IPAM. The domain name is taken from the name field of the resource record. If the domain does not exist in Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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.cisco.com. 600 SRV 0 100 400 pdc.sw.cisco.com. The service specification part would be: _ldap._tcp.pdc._msdcs and the domain name part would be: sw.cisco.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 Cisco Prime Network Registrar 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.cisco.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.cisco.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.cisco.com`.

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

Usage Example 3

This example imports the resource records declared in the zone file *db.hr.cisco.com*. The domain `hr.cisco.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.cisco.com` is created and associated with the server `dns1.sw.cisco.com`.

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

Usage Example 4

This example creates master zones linked to the server `dns1.sw.cisco.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.cisco.com
–m “example.com=external”
```

Usage Example 5

This example creates master zones linked to the server `dns1.sw.cisco.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.cisco.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.cisco.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.cisco.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.

ImportDomainResourceRecord

Overview

The **`ImportDomainResourceRecord`** CLI allows the user to bulk import DNS resource records for a domain into Cisco Prime Network Registrar 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 <adminId>
-p <admin password> -f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

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

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ImportDomainResourceRecord.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the CSV 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

Col	Field	Accepted Values	Required
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

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 Cisco Prime Network Registrar IPAM. However, during the initial setup of the system the queried information from these tasks can be used to populate the original Cisco Prime Network Registrar 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 <adminId> -p <admin password> -t <taskId> [-r <rejects file>][-e <error messages>]

[-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportElementSnapshot.sh –u <adminId> -p <admin password> -t <taskId>
[-r <rejects file>] [-e <error messages>][-v] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ImportElementSnapshot.cmd –u <adminId> -p <admin password> -t <taskId> [-r <rejects
file>] [-e <error messages>] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-i <IP Address>	No	Use to ignore IP Addresses already in use by a defined device. Multiple addresses are separated by a space.
-?	No	Print help
-t <taskId>	Yes	The Id of the "Discover Router Subnets" or "Global Synchronization of Network Elements" task.
-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 blocks and interfaces discovered by task 12345, places into the *elementsnapshot.reject* file any records that could not be imported, and reports errors to the *importerrors.txt* file.

```
$INCHOME/etc/cli/ImportElementSnapshot.sh –u joe –p joepwd –t 12345 -r
elementsnapshot.reject –e importerrors.txt
```

ImportGalaxyDomain

Overview

The **`ImportGalaxyDomain`** CLI allows the user to bulk import Galaxy Domains into Cisco Prime Network Registrar IPAM.

Usage

Direct

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

Via command script (Unix)

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

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ImportGalaxyDomain.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the CSV 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 Cisco Prime Network Registrar 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

Col	Field	Accepted Values	Required
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 Cisco Prime Network Registrar IPAM.

Usage

Direct

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

Via command script (Unix)

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

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ImportNetElement.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the CSV 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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

ImportNetElementInterface

Overview

The **`ImportNetElementInterface`** CLI allows the user to bulk import network element interfaces into Cisco Prime Network Registrar IPAM.

Usage

Direct

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

Via command script (Unix)

```
$INCHOME/etc/cli/ImportNetElementInterface.sh –u <adminId> -p <admin password>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ImportNetElementInterface.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-v} [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the CSV 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar IPAM.

Usage

Direct

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

Via command script (Unix)

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

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ImportNetService.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameter

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the CSV 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
C	Type	The type of Network Service. Accepted value is **dhcp**. If this column is left blank, **dhcp** is assumed.	No

Col	Field	Accepted Values	Required
D	Product name	The Network Service product name. This must be a value already defined in Cisco Prime Network Registrar IPAM, for exadmple**, INS DHCP** or **CNR DHCP**.	Yes
E	Agent name	The name of the Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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.	Yes
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.	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'. If the collection method is cnrsdk, this field is not used.	Yes
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 Path/Executable of NRCD command, the NRCMD user id, the NRCMD password and the Cluster Name. For example, /opt/nwreg2/local/usrbin|myuserid|mypass|cluster1	No

Col	Field	Accepted Values	Required
		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 Cisco Prime Network Registrar IPAM by applying a pre-defined Server Template.

Using the Cisco Prime Network Registrar 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 <adminId>
-p <admin password> -f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

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

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ImportNetServiceWithTemplate.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the CSV 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 or IPv6 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 Cisco Prime Network Registrar IPAM Agent that is responsible for contacting this Network Service as defined in System → Agents.	Yes

ImportRootBlock

Overview

The **`ImportRootBlock`** CLI allows the user to bulk import root blocks into Cisco Prime Network Registrar IPAM.

Usage

Direct

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

Via command script (Unix)

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

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ImportRootBlock.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the CSV 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 container that will hold the block. Names can be in either short or long format. Short format example: **Dallas**. Long format example: **Cisco Prime Network Registrar IPAM/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 Cisco Prime Network Registrar IPAM.	No
J	Allocation Reason	The name of a pre-existing Allocation Reason. If Allocation Reason is not currently in Cisco Prime Network Registrar 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 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.

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 Cisco Prime Network Registrar IPAM. However, during the initial setup of the system the queried information from these tasks can be used to populate the original Cisco Prime Network Registrar 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 <adminId> -p <admin password> -t <taskId> [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ImportServiceSnapshot.sh –u <adminId> -p <admin password> -t <taskId>
[-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ImportServiceSnapshot.cmd –u <adminId> -p <admin password> -t <taskId> [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	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 Cisco Prime Network Registrar IPAM, and to update existing DNS zones.

Usage

Direct

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

Via command script (Unix)

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

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ImportZone.cmd –u <adminId> -p <admin password> -f <import filename> [-r <rejects file>]
[-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the CSV 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
```

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**.	Yes
C	Domain Name	Domain name already defined to Cisco Prime Network Registrar IPAM	Yes

Col	Field	Accepted Values	Required
D	Domain Type	Domain type name already defined to Cisco Prime Network Registrar 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**.	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 Cisco Prime Network Registrar 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 **false**.	No
I	MNAME	For zone type Master only, specify the alternative MNAME. If not specified, no MNAME will be used. This field is not valid for other zone types.	No
J	Allow Zone Transfers to DNS Listener	Accepted values are **true** or **false**. If not specified, defaults to **true**.	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
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	Name of the Galaxy to which to assign this zone. Future Use	No, unless Server is not specified.

ImportZoneResourceRecord

Overview

The **`ImportZoneResourceRecord`** CLI allows the user to bulk import DNS resource records for a zone into Cisco Prime Network Registrar 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 <adminId> -p <admin password>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

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

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ImportZoneResourceRecord.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the CSV 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

Col	Field	Accepted Values	Required
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

Exports

Export CLIs allow you to retrieve and filter information from Cisco Prime Network Registrar 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.

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 <user> -p <pwd> [-n name] [-i <ipaddress>] [-j <interface] [-d <user defined field
name=value>] [-r <ipaddressrange>] [-s <status>] [-b <block>] [-c <container>] [-t
<blocktype>] [-f <export-file>] [-pb <parentBlock>] [-pc <parentContainer>] [-v] [-h] [-
l] [-?]
```

Via command script (Unix)

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

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ExportChildBlock.cmd -u <user> -p <pwd> [-n name] [-i <ipaddress>] [-j <interface]
[-d <user defined field name=value>][-r <ipaddressrange>][-s <status>] [-b <block>]
[-c <container>] [-t <blocktype>][-f <export-file>] [-pb <parentBlock>]
[-pc <parentContainer>] [-v] [-h] [-l] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	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 " ".
-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.
-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. Specify the name of a specific container or use the wildcard character '*' to perform a partial match on the container name.

Parameter	Required	Description
-s <block status>	No	Filter. Specify the block status. {*free*, *aggregate*, *reserved*, *subnet*, *fullyassigned*}. Note: if blocks of status '*free*' are desired, the –l option must be used.
-i <IP address>	No	Filter. Specify an IP address which falls within the start and end address of a block.
-j <Interface>	No	Filter. Specify the name of an interface to which the block must be attached.
-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.
-l (--includeFreeBlocks)	No	Flag. By default, the list of exported child blocks will not include the free blocks which are maintained by Cisco Prime Network Registrar IPAM. Include this Boolean option for the free blocks to be included in the export.
-h (--recurse)	No	Only valid when –pb is specified. When present, recursively exports child blocks of the named parent block.

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: **Cisco Prime Network Registrar IPAM/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
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

Col	Field	Accepted Values	Required
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 Cisco Prime Network Registrar 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, or offset from the beginning, for the interface IP address. 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).	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”.	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 Cisco Prime Network Registrar IPAM that should apply to this subnet. **Valid only for Deployed blocks.**	No.

Col	Field	Accepted Values	Required
S	DHCP Policy Set	The name of a DHCP Policy Set defined within Cisco Prime Network Registrar 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.cisco.com.|dmz.com./External In this example, hr.cisco.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.

Example

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

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

Export via parent block name parameter eexample

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

```
$INCHOME/etc/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/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` contained in container `/Cisco Prime Network Registrar IPAM/Canada/North`.

```
$INCHOME/etc/ExportChildBlock.sh –u joe –p joepwd –f rbexport.csv –pb “172.16.0.0/23”
–pc “/Cisco Prime Network Registrar IPAM/Canada/North”
```

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 <adminId> -p <admin password> -f <export filename> [-n containerName]
[-d udfName=value] [-fp] [-?] [-v]
```

Via command script (Unix)

```
$INCHOME/etc/ExportContainer.sh –u <adminId> -p <admin password> -f <export filename>
[-i ipaddress] [-n containerName] [-d udfName=value] [-fp] [-?] [-v]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ExportContainer.cmd –u <adminId> -p <admin password> -f <export filename>
[-n containerName] [-d udfName=value] [-fp] [-?] [-v]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	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.
-fp	No	Populates the parent container field using the long format, for example: Cisco Prime Network Registrar IPAM/Texas/Dallas
-v	No	Produces verbose output.

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
Col	Field	Accepted Values
A	Container Name	The name of the container.

Col	Field	Accepted Values
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	The name of a pre-existing information template to be associated with this container.
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".

Example

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

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

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

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

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 <adminId> -p <admin password> -f <export filename> [-i ipaddress] [-n deviceName]
[-d udfName=value] [-m domainName] [-r ipaddress1/ipaddress2] [-e devicetype]
[-b blockName] [-c containerName] [-t blocktype] [-?] [-v]
```

Via command script (Unix)

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

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ExportDevice.cmd –u <adminId> -p <admin password> -f <export filename> [-i ipaddress]
[-n deviceName] [-d udfName=value] [-m domainName] [-r ipaddress1/ipaddress2]
[-e devicetype] [-b blockName] [-c containerName] [-t blocktype] [-?] [-v]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	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 the device is located in.
-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.
-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 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.
-b [blockName]	No	Filer. 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.

Parameter	Required	Description
-v	No	Produces verbose output.

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.	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 '|'	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

Examples

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

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

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 <adminId>
-p <admin password> -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/ExportDeviceResourceRecord.sh –u <adminId> -p <admin password>
-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) from the %INCHOME%/etc/cli directory

```
ExportDeviceResourceRecord.cmd –u <adminId> -p <admin password> -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 <pwd>	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 the device is located in.
-d <udfName=value>	No	Filter. Allows you to filter export based the user defined name and value the device owns.
-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.
-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.

Parameter	Required	Description
-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	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.

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=Cisco Prime Network Registrar
IPAM/Texas/Dallas,owner=switch00026,domain=mycompany.com,domainType=Default,resourceRecTy
pe=A:owner=switch00026,resourceRecType=A,TTL=2400,data=10.0.0.24,domain=mycompany.com,dom
ainType=Default,comment=
ipAddress=10.0.0.24,container=Cisco Prime Network Registrar
IPAM/Texas/Dallas,owner=24.0.0,domain=10.in-addr.arpa,domainType=Default,resourceRecType=
PTR:owner=24.0.0,resourceRecType=PTR,TTL=2400,data=switch00026.mycompany.com,domain=10.in
-addr.arpa,domainType=Default,comment=
ipAddress=10.0.0.24,container=Cisco Prime Network Registrar
IPAM/Texas/Dallas,owner=switch00026,domain=mycompany.com,domainType=External,resourceRecT
ype=A:owner=switch00026,resourceRecType=A,TTL=2400,data=10.0.0.24,domain=mycompany.com,do
mainType=External,comment=
ipAddress=10.0.0.24,container=Cisco Prime Network Registrar
IPAM/Texas/Dallas,owner=24.0.0,domain=10.in-addr.arpa,domainType=External,resourceRecType
=PTR:owner=24.0.0,resourceRecType=PTR,TTL=2400,data=switch00026.mycompany.com,domain=10.i
n-addr.arpa,domainType=External,comment=
```

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

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.

Usage

Direct

```
$INCHOME/jre/bin/java –cp $CLASSPATH com.diamondip.netcontrol.cli.ExportNetElementCLI
–u <adminId> -p <admin password> -f <export filename> [Name=<NetElement Name>]
[Address=<IP Address or host name>] [Vendor=<Network Element vendor>]
[Model=<Network Element model>] [Type=<Element type>] [GlobalSync=<Y|N>]
[Agent=<Agent Name>] [Container=<Container Name>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/ExportNetElement.sh –u <adminId> -p <admin password> -f <export filename>
[Name=<NetElement Name>] [Address=<IP Address or host name>]
[Vendor=<Network Element vendor>] [Model=<Network Element model>] [Type=<Element type>]
[GlobalSync=<Y|N>] [Agent=<Agent Name>] [Container=<Container Name>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ExportNetElement.cmd –u <adminId> -p <admin password> -f <export filename>
[Name=<NetElement Name>] [Address=<IP Address or host name>]
[Vendor=<Network Element vendor>] [Model=<Network Element model>] [Type=<Element type>]
[GlobalSync=<Y|N>] [Agent=<Agent Name>] [Container=<Container Name>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	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.
-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.
Name=<NetElement 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.
Vendor=<Network Element vendor>	No	Filter. Allows you to filter export based on specific fields. See file format for accepted values.
Model=<Network Element model>	No	Filter. Allows you to filter export based on specific fields. See file format for accepted values.
Type=<Element type>	No	Filter. Allows you to filter export based on specific fields. See file format for accepted values.
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.

Parameter	Required	Description
Container=<Container Name>	No	Filter. Allows you to filter export based on specific fields. See file format for accepted values.

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 Cisco Prime Network Registrar 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
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

Example

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

```
$INCHOME/etc/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/ExportNetElement.sh –u joe –p joepwd
–f neexport.csv GlobalSync=N Vendor=Cisco
```

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 <adminId> -p <admin password> -f <export filename> [-name <NetService Name>]
[-address <IP Address or host name>] [-type <NetService Type>] [-product <Product name>]
[-agent <Agent name>] [-globalSync <Y|N>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/ExportNetService.sh –u <adminId> -p <admin password> -f <export filename>
[-name <NetService Name>] [-address <IP Address or host name>] [-type <NetService Type>]
[-product <Product name>] [-agent <Agent name>] [-globalSync <Y|N>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ExportNetService.cmd –u <adminId> -p <admin password> [-f <export filename>] [-name
<NetService Name>] [-address <IP Address or host name>] [-type <NetService Type>]
[-product <Product name>] [-agent <Agent name>] [-globalSync <Y|N>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	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 <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".
-globalSync <Y|N>	No	Filter. Allows you to filter export based on specific fields. See file format for accepted values.

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 Cisco Prime Network Registrar IPAM, for example**, INS DHCP** or **CNR DHCP**.	Yes
E	Agent name	The exact name of the Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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
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.	

Col	Field	Accepted Values	Required
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/nwreg2/local/usrbin|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.

Example

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

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

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 <adminId>
-p <admin password> -f <export filename> -m <domain>
-a <domaintype> -q <requesting admin> -x <action> [-?] [-v]
```

Via command script (Unix)

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

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ExportResourceRecordPendingApproval.cmd –u <adminId> -p <admin password>
-f <export filename> -m <domain> -a <domaintype> -q <requesting admin> -x <action> [-?]
[-v]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	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.

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 Cisco Prime Network Registrar 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.

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.

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 <adminId> -p <admin password> -f <export filename> -m <domain>
-a <domaintype> -x <action> [-?] [-v]
```

Via command script (Unix)

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

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ExportResourceRecordPendingApprovalStatus.cmd –u <adminId> -p <admin password>
-f <export filename> -m <domain> -a <domaintype> -x <action> [-?] [-v]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	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.

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 Cisco Prime Network Registrar 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.

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.
```

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 <adminId> -p <admin password> -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/ExportRootBlock.sh –u <adminId> -p <admin password>
-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) from the %INCHOME%/etc/cli directory

```
ExportRootBlock.cmd –u <adminId> -p <admin password> -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 <pwd>	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. Specify the name of a specific container or use the wildcard character '*' to perform a partial match on the container name.
-i <IP address>	No	Filter. Specify an IP address which falls within the start and end address of a block.
-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.

Parameter	Required	Description
-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.

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: **Cisco Prime Network Registrar IPAM/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
I	Organization Id	The organization id for the Regional Internet Registry this space was obtained from. This id must be predefined in Cisco Prime Network Registrar IPAM.	No
J	Allocation Reason	The name of a pre-existing Allocation Reason. If Allocation Reason is not currently in Cisco Prime Network Registrar 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

Col	Field	Accepted Values	Required
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 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.

Example

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

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

Deletions

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, Cisco Prime Network Registrar 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 <adminId> -p <admin password> -f <import filename> [-r <rejects file>]
[-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteAggregateBlock.sh –u <adminId> -p <admin password>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
DeleteAggregateBlock.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the CSV 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	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: Cisco Prime Network Registrar IPAM/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

Col	Field	Accepted Values	Required
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	The specific address, or offset from the beginning, for the interface IP address. 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). An offset of 1 is assumed if none is specified.	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

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
```

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 <adminId> -p <admin password> [-f <import filename>] [-r <rejects file>]
[-e <error messages>] [-b <block Name>] [-c <container name>] [-?] [-x]
```

Via command script (Unix)

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

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
DeleteBlock.cmd –u <adminId> -p <admin password> [-f <import filename>]
[-r <rejects file>] [-e <error messages>] [-b <block Name>]
[-c <container name>] [-?][-x]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	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 <import 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.

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
B	Container Name	The name of the container holding the block.	No

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
```

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 <adminId> -p <admin password> [-f <import filename>] [-r <rejects file>]
[-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteContainer.sh –u <adminId> -p <admin password>
[-f <import filename>] [-r <rejects file>] [-e <error messages>]  [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
DeleteContainer.cmd –u <adminId> -p <admin password> [-f <import filename>]
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import 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.

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

Example

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

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

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.

Usage

Direct

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

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteDevice.sh –u <adminId> -p <admin password>
[-f <import filename>] [-r <rejects file>] [-e <error messages>]  [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
DeleteDevice.cmd –u <adminId> -p <admin password> [-f <import filename>]
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import 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.

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

Col	Field	Accepted Values	Required
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.

Example

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

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

DeleteDeviceInterface

Overview

The **`DeleteDeviceInterface`** CLI allows the user to remove device interfaces from devices in the system. This CLI enables you to specify a file containing a list of device interfaces to delete.

Usage

Direct

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

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteDeviceInterface.sh –u <adminId> -p <admin password>
[-f <import filename>] [-r <rejects file>] [-e <error messages>]  [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
DeleteDeviceInterface.cmd –u <adminId> -p <admin password> [-f <import filename>]
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import 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 Examples

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), but only one IP Address is required to identify the device.	Yes

DeleteDeviceInterface

Col	Field	Accepted Values	Required
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.
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 '|'.	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 Cisco Prime Network Registrar IPAM.

Usage

Direct

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

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteDeviceResourceRecord.sh –u <adminId> -p <admin password>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
DeleteDeviceResourceRecord.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	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.

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.

Col	Field	Accepted Values	Required
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 Cisco Prime Network Registrar IPAM.

Usage

Direct

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

Via command script (Unix)

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

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
DeleteDomainResourceRecord.cmd –u <adminId> -p <admin password> -f <delete filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	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.

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.

Usage

Direct

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

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteNetElement.sh –u <adminId> -p <admin password>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
DeleteNetElement.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import 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.

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.

DeleteNetElement

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 Cisco Prime Network Registrar 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 <adminId> -p <admin password>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteNetElementInterface.sh –u <adminId> -p <admin password>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
DeleteNetElementInterface.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-v} [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import 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.

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.

DeleteNetElementInterface

Col	Field	Accepted Values	Required
A	Name	The name of a Network Element already defined to Cisco Prime Network Registrar 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 <adminId> -p <admin password> -f <import filename> [-r <rejects file>]
[-e <error messages>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteNetService.sh –u <adminId> -p <admin password>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
DeleteNetService.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import 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.

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".

DeleteNetService

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

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 <adminId> -p <admin password> [-t <idlist> | –r <retain> | -d <date>] [-v] [-?]
```

Via command script (Unix)

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

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
DeleteTask.cmd –u <adminId> -p <admin password>
 [-t "<idlist>" | –r <retain> | -d <date>] [-v]  [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	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.

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.

DeleteZoneResourceRecord

Overview

The **`DeleteZoneResourceRecord`** CLI allows the user to delete DNS resource records for a zone from Cisco Prime Network Registrar IPAM.

Usage

Direct

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

Via command script (Unix)

```
$INCHOME/etc/cli/DeleteZoneResourceRecord.sh –u <adminId> -p <admin password>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
DeleteZoneResourceRecord.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	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.

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

Col	Field	Accepted Values	Required
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

Tasks

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 <user> -p <password> [-n <DHCP server name>|-i <DHCP server IP address>][-a][-f][-?]
[-v]
```

Via command script (Unix)

```
$INCHOME/etc/DhcpConfigurationTask.sh –u <user> -p <password>
[-n <DHCP server name>|-i <DHCP server IP address>][-a][-f][-?][-v]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
DhcpConfigurationTask.cmd –u <user> -p <password>
[-n <DHCP server name>|-i <DHCP server IP address>][-a][-f][-?][-v]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-i <IP Address>	Yes, unless –n specified	IP Address of the DHCP Server to configure.
-n <Name>	Yes, unless –i specified	Name of Network Service (DHCP server) to configure.
-a	No	If set, the task will stop if errors are detected during the configuration generation.
-f	No	If set, any associated failover servers will also have their configurations generated and deployed.
-v	No	Verbose

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/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 <adminId> -p <admin password>
(-n <network service name> or -i <network service ip address>) [-?]
```

Via command script (Unix)

```
$INCHOME/etc/DHCPUtilization.sh –u <adminId> -p <admin password>
(-n <network service name> or -i <network service ip address>) [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
DHCPUtilization.cmd –u <adminId> -p <admin password>
(-n <network service name> or -i <network service ip address>) [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	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 Service (DHCP server) to discover. Required only if IP Address or FQDN not specified.

Example

This example creates a task on the queue to perform a DHCP utilization collection for the `dhcp5` DHCP server.

```
$INCHOME/etc/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 Cisco Prime Network Registrar IPAM.

Usage

Direct

```
$INCHOME/jre/bin/java –cp $CLASSPATH com.diamondip.netcontrol.cli.DiscoverNetElementCLI
–u <adminId> -p <admin password> (-n <network element name> or -i <netelement address>)
[-?]
```

Via command script (Unix)

```
$INCHOME/etc/DiscoverNetElement.sh –u <adminId> -p <admin password>
[-n <network element name>] (-n <network element name> or -i <netelement address>) [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
DiscoverNetElement.cmd –u <adminId> -p <admin password>
(-n <network element name> or -i <netelement address>) [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	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.

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/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/DiscoverNetElement.sh –u joe –p joepwd –i corerouter1.mycompany.com
```

This example creates a task to perform a Discover for the Cisco Prime Network Registrar IPAM network element named `router3`.

```
$INCHOME/etc/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 <user> -p <password> [-n <Name>|-i <IP address>][-a][-c][-d][-g][-k] [-s]
[-w <View>][-z <Zone>][-?][-v]
```

Via command script (Unix)

```
$INCHOME/etc/DnsConfigurationTask.sh –u <user> -p <password>
[-n <Name>|-i <IP address>][-a][-c][-d][-g][-k] [-s] [-w <View>][-z <Zone>][-?][-v]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
DnsConfigurationTask.cmd –u <user> -p <password> [-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 <pwd>	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 IP Control. 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.
-v	No	Verbose
-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.

Parameter	Required	Description
-z <Zone>	No, unless –w specified	Zone name. Set this option to generate a configuration for this selected zone, instead of the whole server.

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/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/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/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/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/DnsConfigurationTask.sh –u joe –p joepwd –c –d –n dns1 –z acme.com
```

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 ModifyBlockCLI 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 <adminId> -p <admin password> -t <#><period> [-?]
```

Via command script (Unix)

```
$INCHOME/etc/GlobalRollup.sh –u <adminId> -p <admin password> -t <#><period> [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
GlobalRollup.cmd –u <adminId> -p <admin password> -t <#><period> [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	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).

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/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/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 <adminId> -p <admin password> -t <type> [-?]
```

Via command script (Unix)

```
$INCHOME/etc/GlobalSync.sh –u <adminId> -p <admin password> -t <type> [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
GlobalSync.cmd –u <adminId> -p <admin password> -t <type> [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-t <type>	Yes	Indicates the type of Global Sync to perform. Acceptable values are **netelement** or **netservice**.

Example

This example creates a task on the queue to perform a Global Synchronization of all network elements that are flagged in Cisco Prime Network Registrar IPAM for inclusion in the Global Sync process.

```
$INCHOME/etc/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 Cisco Prime Network Registrar IPAM for inclusion in the Global Sync process.

```
$INCHOME/etc/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 <adminId> -p <admin password> -t <task number> [-v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/TaskStatus.sh –u <adminId> -p <admin password> -t <task number> [-v] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
TaskStatus.cmd –u <adminId> -p <admin password> -t <type> [-v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	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.

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>
```

Example

This example queries task `37` and returns a one-word string indicating the status:

```
INPROGRESS
```

```
$INCHOME/etc/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/TaskStatus.sh –u joe –p joepwd –t 42 -v
```

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 com.diamondip.netcontrol.cli.DetachBlockCLI
–u <adminId> -p <admin password> -f <import filename> [-r <rejects file>]
[-e <error messages>] [–v] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/DetachBlock.sh –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [–v] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
DetachBlock.cmd –u <adminId> -p <admin password> -f <import filename> [-r <rejects file>]
[-e <error messages>] [–v] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import 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.

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: Cisco Prime Network Registrar IPAM/Texas/Dallas. Long format eliminates ambiguity in cases where there are duplicate container names.	Yes

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 <adminId> -p <admin password> [-r <rejects file>] [-e <error messages>]
[-b <block Name>] [-c <container name>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/JoinBlock.sh –u <adminId> -p <admin password>  [-r <rejects file>]
[-e <error messages>] [-b <block Name>] [-c <container name>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
JoinBlock.cmd –u <adminId> -p <admin password> [-r <rejects file>] [-e <error messages>]
[-b <block Name>] [-c <container name>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	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.

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 <adminId> -p <admin password> -f <import filename> [-r <rejects file>] [-e <error
messages>] [-?]
```

Via command script (Unix) from the $INCHOME/etc/cli directory

```
ModifyAddrpool.sh –u <adminId> -p <admin password> -f <update filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ModifyAddrpool.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the CSV file to import. 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.

Output

If successful, the CLI updates the address pools per the input file and exits.

Example

```
$INCHOME/etc/cli/ModifyAddresspool.sh –u joe –p joepwd -f updateaddrpools.txt
-r updateaddrpools.reject –e updateaddrpool.err
```

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
Address Pool Type	type	One of "Dynamic DHCP", "Automatic DHCP", "Static", "Reserved".	No
Name	name	Address Pool name. Defaults to "Start Address-End Address"	No
Share Name	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

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=
```

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 <adminId> -p <admin password> -f <import filename> [-r <rejects file>]
[-e <error messages>] [-?]
```

Via command script (Unix) from the $INCHOME/etc/cli directory

```
ModifyBlock.sh –u <adminId> -p <admin password> -f <update filename> [-r <rejects file>]
[-e <error messages>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ModifyBlock.cmd –u <adminId> -p <admin password> -f <import filename> [-r <rejects file>]
[-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import 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.

Output

If successful, the CLI updates the blocks per the input file and exits.

Example

```
$INCHOME/etc/cli/ModifyBlock.sh –u joe –p joepwd -f updateblocks.txt
-r updateblocks.reject –e updateblock.err
```

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
Block Name	blockName	The Name of the block	Yes / Yes

Field	Attribute Name	Accepted Values	Locator Field/ Modify?
Block Size	blockSize	The CIDR size of the block	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
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
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
Address Allocation Details	addrDetails	Define attributes of any address allocations of the specified allocation template. See below for syntax.	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 =
```

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"
```

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=Cisco DHCP Subnet Template Policy Set,
DNSServers=ServerABC,defaultGateway=192.80.0.1,failoverDHCPServer=DHCPFailoverServer,forw
ardDomains=test.com|example.com,forwardDomainTypes=Default|Internal,primaryDHCPServer=DHC
PServer,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 Cisco Prime Network Registrar IPAM that should apply to this subnet.	No
DHCP Policy Set	DHCPPolicySet	The name of a DHCP Policy Set defined within Cisco Prime Network Registrar 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
Default Gateway	defaultGateway	The default gateway address for this subnet. This address is supplied to DHCP clients on this subnet.	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

Field	Attribute Name	Accepted Values	Locator Field
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
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,
sharename=poolsharename}
```

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?	offsetFromBeginningOfSubnet	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
Shared Network Name	sharename	The name used to link address pools together.	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 <adminId> -p <admin password> -f <import filename> [-r <rejects file>]
[-e <error messages>] [-?]
```

Via command script (Unix) from the $INCHOME/etc/cli directory

```
ModifyContainer.sh –u <adminId> -p <admin password> -f <update filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ModifyContainer.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f < 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.

Output

If successful, the CLI updates the containers per the input file and exits.

Example

```
$INCHOME/etc/cli/ModifyContainer.sh –u joe –p joepwd -f updatecontainers.txt
-r updatecontainers.reject –e updatecontainers.err
```

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: **Cisco Prime Network Registrar IPAM/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	The name of a pre-existing information template to be associated with this container.	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

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=San Jose”
```

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=|ne
wTemplate|newTemplate,allowedDeviceTypes=Printer|Router|Switch,deviceInfoTemplates=xTempl
ate|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 <adminId> -p <admin password> -f <import filename> [-r <rejects file>]
[-e <error messages>] [-?]
```

Via command script (Unix) from the $INCHOME/etc/cli directory

```
ModifyDevice.sh –u <adminId> -p <admin password> -f <update filename> [-r <rejects file>]
[-e <error messages>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ModifyDevice.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the CSV file to import. 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.

Output

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

Example

```
$INCHOME/etc/cli/ModifyDevice.sh –u joe –p joepwd -f updatedevices.txt
-r updatedevices.reject –e updatedevices.err
```

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.

ModifyDevice

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=San Jose”
```

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.

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 <adminId> -p <admin password>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix) from the $INCHOME/etc/cli directory

```
ModifyDeviceResourceRecord.sh –u <adminId> -p <admin password> -f <update filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ModifyDeviceResourceRecord.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the CSV file to import. 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.

Output

If successful, the CLI updates the device resource records per the input file and exits.

Example

```
$INCHOME/etc/cli/ModifyDeviceResourceRecord.sh –u joe –p joepwd
-f updateeviceresourcerecs.txt -r updatedeviceresourcerecs.reject
–e updatedeviceresourcerecs.err
```

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,resour
ceRecType=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,resour
ceRecType=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,resour
ceRecType=A:comment=
```

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 <adminId> -p <admin password> -f <import filename> [-r <rejects file>] [-e <error
messages>] [-?]
```

Via command script (Unix) from the $INCHOME/etc/cli directory

```
ModifyDhcpServer.sh –u <adminId> -p <admin password> -f <update filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ModifyDhcpServer.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The CSV file to import. 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.

Output

If successful, the CLI updates DHCP servers per the input file and exits.

Example

```
$INCHOME/etc/cli/ModifyDhcpServer.sh –u joe –p joepwd -f updatedhcp.txt
-r updatedhcp.reject –e updatedhcp.err
```

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.

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 Cisco Prime Network Registrar IPAM.	No
Agent	agent	The name of an agent defined in Cisco Prime Network Registrar 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	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	True or False	No
DHCP Option Set	optionSet	Valid Option Set defined in Cisco Prime Network Registrar IPAM	No
DHCP Policy Set	policySet	Valid Policy Set defined in Cisco Prime Network Registrar IPAM.	No
DHCP Client Classes	clientClasses	Valid DHCP Client Classes defined in Cisco Prime Network Registrar 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

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
```

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 <adminId> -p <admin password>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix) from the $INCHOME/etc/cli directory

```
ModifyDomainResourceRecord.sh –u <adminId> -p <admin password> -f <update filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ModifyDomainResourceRecord.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The name of the CSV file to import. 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.

Output

If successful, the CLI updates the domain resource records per the input file and exits.

Example

```
$INCHOME/etc/cli/ModifyDomainResourceRecord.sh –u joe –p joepwd
-f updatedomainresourcerecs.txt -r updatedomainresourcerecs.reject
–e updatedomainresourcerecs.err
```

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”.

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 <adminId> -p <admin password>
-f <import filename> [-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Unix) from the $INCHOME/etc/cli directory

```
ModifyNetElementInterfaceCLI.sh –u <adminId> -p <admin password> -f <update filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ModifyNetElementInterfaceCLI.cmd –u <adminId> -p <admin password> -f <import filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <import filename>	Yes	The CSV file to import. 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.

Output

If successful, the CLI updates Network Element Interfaces per the input file and exits.

Example

```
$INCHOME/etc/cli/ ModifyNetElementInterfaceCLI.sh –u joe –p joepwd -f updatenei.txt
-r updatenei.reject –e updatenei.err
```

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 <adminId> -p <admin password> -f < update filename> [-r <rejects file>]
[-e <error messages>] [-?]
```

Via command script (Unix) from the $INCHOME/etc/cli directory

```
ModifyPendingApproval.sh –u <adminId> -p <admin password> -f <update filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
ModifyPendingApproval.cmd –u <adminId> -p <admin password> -f < update filename>
[-r <rejects file>] [-e <error messages>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	Yes	Password
-?	No	Print help
-f <modify 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.

Output

If successful, the CLI approves and rejects changes per the input file and exits.

Example

```
$INCHOME/etc/cli/ModifyPendingApproval.sh –u joe –p joepwd -f pendingapprovals.csv
-r pendingapprovals.reject –e pendingapprovals.err
```

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”
```

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.

Usage

Direct

```
$INCHOME/jre/bin/java –cp $CLASSPATH com.diamondip.ipcontrol.cli.SplitBlockCLI
–u <adminId> -p <admin password> [-r <rejects file>] [-e <error messages>]
[-b <block Name>] [-c <container name>] [-t <target start address>] [-s <target size>]
[-q <equal sizes>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/cli/ SplitBlockCLI.sh –u <adminId> -p <admin password>  [-r <rejects file>]
[-e <error messages>] [-b <block Name>] [-c <container name>] [-t <target start address>]
[-s <target size>] [-q <equal sizes>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
SplitBlockCLI.cmd –u <adminId> -p <admin password> [-r <rejects file>]
[-e <error messages>] [-b <block Name>] [-c <container name>] [-t <target start address>]
[-s <target size>] [-q <equal sizes>] [-?]
```

Parameters

Parameter	Required	Description
-u <userId>	Yes	User Id
-p <pwd>	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.

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 <adminId> -p <admin password> -b <blockaddress> –d <devicetype> [-h <hostname>]
[-r <rr flag>] [-?]
```

Via command script (Unix)

```
$INCHOME/etc/UseNextReservedIPAddress.sh –u <adminId> -p <admin password>
-b <blockaddress> –d <devicetype> [-h <hostname>] [-r <rr flag>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
UseNextReservedIPAddress.cmd –u <adminId> -p <admin password> -b <blockaddress>
–d <devicetype> [-h <hostname> [-r <rr flag>] [-?]
```

Parameters

Parameter	Required	Description
-u <user id>	Yes	User Id
-p <password>	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

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/DhcpRelease.sh [–s <server>] -m <macaddr> -i <ipaddr> [-f <filename>] [-?]
```

Via command script (Windows) from the %INCHOME%/etc/cli directory

```
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.

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.

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
```

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 Cisco Prime Network Registrar IPAM platform has different behavior with respect to running the CLI locally.

- **Solaris** – not supported, CLI must be run remotely.
- **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.

Application Program Interfaces (API)

Using the API

Cisco Prime Network Registrar 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

Cisco Prime Network Registrar 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 Cisco Prime Network Registrar IPAM login name and password. Cisco Prime Network Registrar 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
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 RR	X	X	X	
Galaxy Domain	X			

Object	Import	Modify	Delete	Export
NetElement	X		X	X
NetElementInterface	X	X	X	
Netservice	X		X	X
RootBlock	X		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 Cisco Prime Network Registrar 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: Cisco Prime Network Registrar IPAM/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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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	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 Cisco Prime Network Registrar IPAM.

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="xsd:string" />
  <element name="interfaceName" nillable="true" type="xsd:string" />
  <element name="ipv6" 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*

DetachBlock

Element	Description and accepted values	Required	Return Code	Faultstring
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: /**Cisco Prime Network Registrar IPAM/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	Ignored	no		
interfaceName	Ignored	no		
ipv6	Ignored	no		
userDefinedFields	Ignored	no		
excludeFromDiscovery	Ignored	no		

ImportAddressPool

Overview

The **`ImportAddressPool`** API enables the web service client to import or modify containers in Cisco Prime Network Registrar 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="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>
```

ImportAddressPool

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	-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”.	Yes	-73	Invalid Address Type
name	Address Pool name. Defaults to “Start Address-End Address”	No		
sharename	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
primaryNet Service	The name of the DHCP server that will serve addresses from this pool	No	-94	DHCP Server not found
failoverNet Service	The name of the failover DHCP server that will serve addresses from this pool	No	-94 -96	DHCP Server not found Failover not valid without a primary
dhcpOptionSe t	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

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: Cisco Prime Network Registrar IPAM/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>

ImportAggregateBlock

Element	Description and accepted values	Required	Return Code	Faultstring
Interface Address	The specific address, or offset from the beginning, for the interface IP address. 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). An offset of 1 is assumed if none is specified.	No	-18 -21	Invalid interface address Invalid offset:
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 o 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 Cisco Prime Network Registrar 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.

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="xsd:string" />
  <element name="interfaceName" nillable="true" type="xsd:string" />
  <element name="ipv6" 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 Cisco Prime Network Registrar 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.	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*
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: **/Cisco Prime Network Registrar IPAM/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.

Element	Description and accepted values	Required	Return Code	Faultstring
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		No validation required
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 address, or offset from the beginning, for the interface IP address. 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 subnet. An offset of 1 is assumed if none is specified.	no	-18 -21 -24	Invalid interface address: *interfaceAddress* Invalid interface offset: *interfaceAddress* Interface address required for
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	Missing interface name Invalid interface name: *interfaceName*
ipv6	True if this is an IPV6 block. If not specified, 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.**	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="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="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
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*
primaryDHCP Server	The name of the DHCP server that will act as primary for this subnet.	no	-94	DHCP Server *server* not found
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*
-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 Cisco Prime Network Registrar 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="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="soapenc:string"/>
  <element name="maintainHistoryRecs" type="xsd:boolean"/>
  <element name="parentName" 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
allowedAllocFromParentBlocktypes	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*
allowedBlockTypes	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*
allowedDeviceTypes	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*
allowedRootBlockTypes	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*
blockTypeInfoTemplates	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	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*
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		No validation required

ImportContainer

Element	Description and accepted values	Required	Return Code	Faultstring
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*
ignoreBlocktypeIn Use	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	The name of the information template to be associated with this container.	no	-1 -78	DB error retrieving information template Invalid Information Template: *templateName*
maintainHistory Recs	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: **Cisco Prime Network Registrar IPAM/Texas/Dallas**. Long format eliminates ambiguity in cases where there are duplicate container names.	yes	-3 -10 -11 -1	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
requireSWIPName BlockTypes	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*

Element	Description and accepted values	Required	Return Code	Faultstring
userDefinedFields	The user defined fields associated with this container, as listed in the container information template 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

ImportDevice

Overview

The **ImportDevice** API enables the web service client to import devices into Cisco Prime Network Registrar 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="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" />
  <element maxOccurs="unbounded" name="interfaces" nillable="true"
           type="tns2:WSInterface"/>
  <element name="excludeFromDiscovery" nillable="true"
type="soapenc:string" />
 </sequence>
</complexType>
<complexType name="WSInterface">
```

```
   <sequence>
     <element name="hwType" nillable="true" type="soapenc:string"/>
     <element name="id" nillable="true" type="soapenc:int"/>
     <element name="macAddress" nillable="true"
type="soapenc:string"/>
     <element name="name" nillable="true" type="soapenc:string"/>
     <element name="sequence" nillable="true" type="soapenc:int"/>
   <element name="ipAddress" nillable="true"
type="impl:ArrayOf_soapenc_string"/>
     <element name="excludeFromDiscovery" nillable="true"
type="soapenc:string" />
   </sequence>
</complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
Domain type	Domain type already defined to Cisco Prime Network Registrar IPAM. If not specified, the "Default" domain type will be used.	No	-59	Domain type not found: *view*
Hwtype	Specify **Ethernet** or **Token Ring**. When hwtype is specified, MACAddress must also be specified.	Yes, for Manual DHCP or if MACAddress is specified.	-74 -72 -76	Hardware type must be specified for address type Manual DHCP Invalid Hardware type: *type* Hardware type must be specified when using MAC Address
addressType	The address type of this device. Accepted values are: **Static**, **Dynamic DHCP**, **Automatic DHCP**, **Manual DHCP**, and **Reserved**.	Yes	-73	Invalid address type: *type*
ipAddress	The IP Address of the device	Yes	-26	Invalid IP Address: *address*
resourceRecordFlag	Whether or not to add resource records for this device. If not specified as **true**, defaults to false.	No		
MACAddress	The hardware MAC address of the device.	Yes, if hwtype is specified.	-77	MAC Address must be specified when using Hardware Type
deviceType	The name of a device type configured in Cisco Prime Network Registrar IPAM.	Yes, if hostname specifies use of naming policy.	-47, -58 -75	Device type not found: *type* Device type required when using naming policy

ImportDevice

Element	Accepted Values	Required	Return Code	Faultstring
domainName	Domain name already defined to Cisco Prime Network Registrar 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
container	The name of the container that contains the device.	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
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
description	A description of the device. Use "\n" to separate lines.	No		
Hostname	Valid host name or **APPLYNAMINGPOLICY**.	Yes		
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
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*
Interfaces	An array of WSInterface structures. Each element in the array corresponds to one interface for a multihomed device. The fields in the WSInterface structure are: name: Interface Name (Required) ipAddress: IP Address (Required) hwType: Same as above macAddress: Same as above sequence: Reserved for future use ID: Reserved for future use	Yes, for mulithome d devices.	-20 -19	Interface name missing. IP Address missing or invalid
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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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	Data Required
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 Cisco Prime Network Registrar 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" type="xsd:boolean"/>
     <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"/>
    </sequence>
   </complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
Agent	The name of the Cisco Prime Network Registrar 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*

ImportDhcpServer

Element	Accepted Values	Required	Return Code	Faultstring
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 Cisco Prime Network Registrar 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	Set to TRUE to enable DDNS updates from this DHCP Server. Defaults to FALSE.	No		
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

Element	Accepted Values	Required	Return Code	Faultstring
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		

Other returnCodes and faultstrings

Return Code	Faultstring
-1	Unable to obtain session or Rollback failed. Additional information will appear in the web service log.

ImportDomainResourceRecord

Overview

The **`ImportDomainResourceRecord`** API enables the web service client to import resource records into Cisco Prime Network Registrar 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="WSDomainResourceRecord">
 <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="domain" nillable="true" type="soapenc:string" />
  <element name="domainType" nillable="true" type="soapenc:string"
/>
  <element name="comment" nillable="true" type="soapenc:string" />
  </sequence>
 </complexType>
```

Element	Accepted Values	Required	Return Code	Faultstring
domainType	Domain type already defined to Cisco Prime Network Registrar 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		

ImportGalaxyDomain

Overview

The **`ImportGalaxyDomain`** API enables the web service client to assign domains to galaxies in Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar IPAM.

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 Cisco Prime Network Registrar IPAM. If not specified, defaults to **Unknown**.	Yes when Model is specified.		no models found for given description: *description* deviceName: *name*, vendorName: *vendor*
3	Model	The model name of the	Yes when		same as above

Element	Description	Accepted Values	Required	Return Code	Faultstring
		Network Element. Model must be predefined in Cisco Prime Network Registrar IPAM. If not specified, defaults to **Unknown**.	Vendor is specified		
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 Cisco Prime Network Registrar 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 Name	SNMP V3 Context name, if needed.	No		
18	V3 Engine ID	SNMP V3 Engine ID, if needed.	No		

ImportNetElementInterface

Overview

The **`ImportNetElementInterface`** API enables the web service client to import Network Element Interfaces into Cisco Prime Network Registrar 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		
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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar IPAM.

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 Cisco Prime Network Registrar IPAM Agent that is responsible for contacting this Network Service.	Yes	-29	Invalid network service agent: *agent*
collectionMethod	The method by which the Cisco Prime Network Registrar IPAM Agent will collect data from the Network Service. Accepted values are **scp** or **ftp**.	Yes	-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).	Yes	-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.	Yes		
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'.	Yes		
vendor	The Network Service product name. This must be a value already defined in Cisco Prime Network Registrar IPAM, for example**, INS DHCP** or **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 NRCMD 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.

ImportRootBlock

Overview

The **`ImportRootBlock`** API enables the web service client to import root blocks into Cisco Prime Network Registrar 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>
```

ImportRootBlock

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 container that will hold the block. Names can be in either short or long format. Short format example: **Dallas**. Long format example: **Cisco Prime Network Registrar IPAM/Texas/Dallas**. Long format eliminates ambiguity in cases where there are duplicate container names.	Yes	-2 -6 -7 -5 -1 -5	Could not find container: *container* Invalid container name Ambiguous container name Container not found Database error Unable to lookup container
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		

Element	Accepted Values	Required	Return Code	Faultstring
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 Cisco Prime Network Registrar IPAM.	No	-84	Invalid Organization Id: *id*
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	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.

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 211) 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.

ModifyBlock

```
   <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="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="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="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		

Element	Description	Modify?	Return Code	Fault String
allocationReason Description	The text entered at creation for the allocation reason.	No.		
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 -21 -65 -177 -182 -189 -207	Allocation Template supported only for In-Use/Deployed blocks Invalid offset Invalid allocation template: *name* Address Pool Template detail overlaps interface address. No domain set, create resource records failed. Default Gateway(s) have been defined in the Subnet Policies. Address pool templates with default gateway option are not allowed. 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		
id	The block ID. This MUST be supplied in order to update the block.	No	-36	Block not found

ModifyBlock

Element	Description	Modify?	Return Code	Fault String
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.	No		
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		
numaddressablehosts	Number of Addressable hosts	No		
numallocatedhosts	Number of Allocated hosts	No		
numassignedhosts	Number of Assigned hosts	No		
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
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

Element	Description	Modify?	Return Code	Fault String
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 211) 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="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="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*
defaultGateway	The default gateway that DHCP clients on this subnet will use. Accepted value is an IP address.	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*
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	The name used to link address pools together.	No/Yes		
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
action	Specify "Approve" or "Reject".	No	-161	Input action null! Invalid action: *action*
reason	The text entered at creation for the allocation reason.	No.		

Element	Description	Modify?	Return Code	Fault String
workflowId	The id of the pending approval request retrieved using an Export*Item*PendingApproval, for example, ExportResourceRecordPendingApproval.	No	-162	Workflow id not found: *id*

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 Cisco Prime Network Registrar IPAM. These services can be used in conjunction with Imports to modify Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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.

getBlock

There are two calls to retrieve blocks:

- **`getBlockByName`**
- **`getBlockByIpAddress`**

Both calls return a **`WSGenericBlock`** data structure. See the **`ModifyBlock`** call (Page 186) for a description of **`WSGenericBlock`**. As their names suggest, they 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.

getBlock

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

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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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

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

Return Codes

Condition	Return Code
Exception reading from db	-2
Domain not found	-60
DNS Resource Record not unique	-125
DNS resource record not found	-136

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

Tasks

Overview

This section explains the web services available for issuing tasks to Cisco Prime Network Registrar IPAM, and for querying the status of Cisco Prime Network Registrar 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

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	Cisco Prime Network Registrar IPAM administrator's user id	Yes
password	Cisco Prime Network Registrar IPAM administrator's password	Yes
elementName	Name of the DHCP server for which Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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>
```

DiscoverNetElement

Part name	Description and accepted values	Required
adminId	Cisco Prime Network Registrar IPAM administrator's user ID	Yes
password	Cisco Prime Network Registrar 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.

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 Cisco Prime Network Registrar 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	Cisco Prime Network Registrar IPAM administrator's user id	Yes
password	Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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	Cisco Prime Network Registrar IPAM administrator's user id	Yes
password	Cisco Prime Network Registrar 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	Cisco Prime Network Registrar IPAM administrator's user id	Yes
password	Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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:

- **`ExportContainer`**
- **`ExportRootBlock`**
- **`ExportChildBlock`**
- **`ExportDevice`**
- **`ExportDeviceResourceRecord`**

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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar IPAM.	Yes when Model is specified.
D	Model	The model name of the Network Element. Model is predefined in Cisco Prime Network Registrar IPAM.	Yes when Vendor is specified.
E	Type	The type of Network Element. Accepted values are **CMTS**, **Router**, **Switch**, or **VPN**.	Yes

Col	Field	Description	Required
F	Global Sync	Whether or not to include this Network Element in the Global Sync process. Value is **true** or **false**.	Yes
G	Agent Name	The exact name of the Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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.
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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar IPAM, for example**, INS DHCP** or **CNR DHCP**.	Yes
E	Agent name	The name of the Cisco Prime Network Registrar 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 Cisco Prime Network	Yes

Col	Field	Accepted Values	Required
		Registrar IPAM Agent collects data from the Network Service. Values are **scp** or **ftp**.	
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/nwreg2/local/usrbin|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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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 237 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 Cisco Prime Network Registrar 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

Element	Description
resultCount	Reserved
internalResultCount	Reserved

ExportRootBlock

Overview

The **`exportRootBlock`** API enables the web service client to issue a request to retrieve a list of Root Blocks from Cisco Prime Network Registrar 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:part name="query" type="soapenc:string">
<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 name 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 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**.

ExportChildBlock

Overview

The **`exportChildBlock`** API enables the web service client to issue a request to retrieve a list of Child Blocks from Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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=/Cisco Prime Network Registrar IPAM/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='/Cisco Prime Network Registrar IPAM/Canada/North'
recursive	Only appied when paretnBlock is supplied. When set to true, all child blocks of the specified parent name recursively will be exported.	recursive=true
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'
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="interfaceAddress" nillable="true" type="soapenc:string" />
  <element name="interfaceName" nillable="true" type="soapenc:string" />
  <element name="ipv6" type="xsd:boolean"/>
  <element name="userDefinedFields" nillable="true" type="impl:ArrayOf_soapenc_string"
/>
  <element name="excludeFromDiscovery" nillable="true" type="soapenc:string"/>
 </sequence>
</complexType>
```

Element	Description and accepted values
SWIPname	SWIP name for this block
allocationReason	The name of a pre-existing Allocation Reason.

ExportChildBlock

Element	Description and accepted values
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.
interfaceAddress	The specific address 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.
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="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="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
DHCPOptionsSet	The name of a previously defined DHCP Options set.
DHCPPolicySet	The name of a previously defined DHCP policy set.

Element	Description and accepted values
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.
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.

ExportContainer

Overview

The **`exportContainer`** API enables the web service client to issue a request to retrieve a list of Containers from Cisco Prime Network Registrar 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: Cisco Prime Network Registrar IPAM/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 170.

ExportDevice

Overview

The **`exportDevice`** API enables the web service client to issue a request to retrieve a list of Devices from Cisco Prime Network Registrar 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'

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 Cisco Prime Network Registrar 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="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" />
  <element maxOccurs="unbounded" name="interfaces" nillable="true"
          type="tns2:WSInterface"/>
  <element name="excludeFromDiscovery" nillable="true" type="soapenc:string" />
 </sequence>
</complexType>
<complexType name="WSInterface">
   <sequence>
     <element name="hwType" nillable="true" type="soapenc:string"/>
     <element name="id" nillable="true" type="soapenc:int"/>
     <element name="macAddress" nillable="true" type="soapenc:string"/>
     <element name="name" nillable="true" type="soapenc:string"/>
     <element name="sequence" nillable="true" type="soapenc:int"/>
     <element name="ipAddress" nillable="true" type="impl:ArrayOf_soapenc_string"/>
     <element name="excludeFromDiscovery" nillable="true" type="soapenc:string" />
   </sequence>
```

```
</complexType>
```

Element	Accepted Values
Domain type	Domain type already defined to Cisco Prime Network Registrar IPAM. If not specified, the “Default” domain type will be used.
hwtype	Specify **Ethernet** or **Token Ring**. When hwtype is specified, MACAddress must also be specified.
addressType	The address type of this device. Accepted values are: **Static**, **Dynamic DHCP**, **Automatic DHCP**, **Manual DHCP**, and **Reserved**.
ipAddress	The IP Address of the device
resourceRecordFlag	Whether or not to add resource records for this device. If not specified as **true**, defaults to false.
MACAddress	The hardware MAC address of the device.
deviceType	The name of a device type configured in Cisco Prime Network Registrar IPAM.
domainName	Domain name already defined to Cisco Prime Network Registrar IPAM
container	The name of the container that contains the device.
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**.
description	A description of the device.
Hostname	Valid host name or **APPLYNAMINGPOLICY**.
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**.
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 multihomed device. The fields in the WSInterface structure are: name: Interface Name (Required) ipAddress: IP Address (Required) hwType: Same as above macAddress: Same as above sequence: Reserved for future use ID: Reserved for future use
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**. For multihomed devices, the flag must be specified for each IP/Interface via the WSInterface structure.

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 Cisco Prime Network Registrar 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	IPAddress=10.0.0.1

Selector	Description	Example
	by Or) Selector type filters.	
IP Address Range	Select device by IP Address range.	IPAddressRange=10.0.0.1-11.0.0.1
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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar IPAM. If not specified, the “Default” domain type will be used.

Element	Accepted Values
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.

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 “cisco.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 Cisco Prime Network Registrar 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.

ExportResourceRecordPendingApproval

```
<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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar IPAM to import the zone data into.

Updates

This section explains the web services available for updating information in Cisco Prime Network Registrar 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

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 -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>
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: Cisco Prime Network Registrar IPAM/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

DeleteBlock

Overview

The **`DeleteBlock`** API enables the web service client to delete blocks from Cisco Prime Network Registrar 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 another in the system.	No, unless block is overlapped.	-99	Container not found

DeleteContainer

Overview

The **`DeleteContainer`** API enables the web service client to delete containers from Cisco Prime Network Registrar 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 Cisco Prime Network Registrar IPAM.

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 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	Faultstring
Domain type	Ignored	No		
Hwtype	Ignored	No		
addressType	Ignored	No		
ipAddress	The IP Address of the device	Yes	-26 -27 -28	Invalid IP Address: *address* IP Address not found IP Address ambiguous
resourceRecord Flag	Ignored	No		
MACAddress	Ignored	No		
deviceType	Ignored	No		
domainName	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
Dupwarning	Ignored	No		
description	Ignored	No		
hostname	Ignored	No		
aliases	Ignored	No		
userDefinedFields	Ignored	No		

DeleteDeviceInterface

Overview

The **DeleteDeviceInterface** API enables the web service client to delete device interfaces from Cisco Prime Network Registrar IPAM.

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		

DeleteDeviceInterface

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) ipAddress: IP Address (Ignored) hwType: (Ignored) macAddress: (Ignored) sequence: (Ignored) ID: (Ignored)	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 Cisco Prime Network Registrar 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

WSDeviceResourceRecord

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>
```

DeleteDeviceResourceRecord

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 overlappin g 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

DeleteDomainResourceRecord

Overview

The **`DeleteDomainResourceRecord`** API enables the web service client to delete resource records from Cisco Prime Network Registrar 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="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
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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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		

Element	Accepted Values	Required	Return Code	Faultstring
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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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:part name="inpNS" type="tns1:WSNetService"/>
   </wsdl:message>
         <wsdl:output message="impl: deleteNetServiceResponse" 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

DeleteTaskByDate

Overview

The **`DeleteTaskByDate`** API enables the web service client to delete tasks from Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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

DeleteZoneResourceRecord

Overview

The **`DeleteZoneResourceRecord`** API enables the web service client to delete resource records from Cisco Prime Network Registrar 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>
```

DeleteZoneResourceRecord

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.

Other Interfaces

Callout Manager

The Callout Manager is a facility within Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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\Diamond IP\InControl\etc` on Windows, or `/opt/incontrol/etc` on Solaris or Linux. Assuming that path is where you installed Cisco Prime Network Registrar IPAM. 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\\Diamond IP\\InControl\\bin\\alertcallout.bat. On Solaris or Linux, an example of the property value would be /opt/incontrol/bin/alertcallout.sh.

Property	Required	Description
block.add	No	The name of the command to execute when a block is added to Cisco Prime Network Registrar IPAM. For example, on Windows, an example of the property value would be c:\\Program Files\\Diamond IP\\InControl\\bin\\ blockadd_callout.bat. On Solaris or Linux, an example of the property value would be /opt/incontrol/bin/blockadd_callout.sh.
block.modify	No	The name of the command to execute when a block is modified within Cisco Prime Network Registrar IPAM. For example, on Windows, an example of the property value would be c:\\Program Files\\Diamond IP\\InControl\\bin\\ blockmodify_callout.bat. On Solaris or 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 Cisco Prime Network Registrar IPAM. For example, on Windows, an example of the property value would be c:\\Program Files\\Diamond IP\\InControl\\bin\\ blockdelete_callout.bat. On Solaris or 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 Cisco Prime Network Registrar IPAM GUI. For example, on Windows, an example of the property value would be c:\\Program Files\\Diamond IP\\InControl\\bin\\ deviceadd_callout.bat. On Solaris or 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 Cisco Prime Network Registrar IPAM GUI. For example, on Windows, an example of the property value would be c:\\Program Files\\Diamond IP\\InControl\\bin\\ devicemodify_callout.bat. On Solaris or Linux, an example of the property value would be /opt/incontrol/bin/devicemodify_callout.sh

Configuration

Property	Required	Description
device.delete	No	The name of the command to execute when a device is deleted within the Cisco Prime Network Registrar IPAM GUI. For example, on Windows, an example of the property value would be c:\\Program Files\\Diamond IP\\InControl\\bin\\ devicedelete_callout.bat. On Solaris or Linux, an example of the property value would be /opt/incontrol/bin/devicedelete_callout.s h
task.complete	No	The name of the command to execute when a task is launched with the Cisco Prime Network Registrar IPAM GUI. For example, on Windows, an example of the property value would be c:\\Program Files\\Diamond IP\\InControl\\bin\\taskcomplete_out. bat. On Solaris or Linux, an example of the property value would be /opt/incontrol/bin/taskcomplete_callout. sh

RIR Template Support

Introduction

Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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

Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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/nc*.)

```
block.add = /opt/nc/etc/callout/addSwip.pl
block.modify = /opt/nc/etc/callout/modifySwip.pl
block.delete = /opt/nc/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

Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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.

Property	Required	Description
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
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 Cisco Prime Network Registrar IPAM Executive system by the system administrator. PERL 5 is NOT supplied with Cisco Prime Network Registrar 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.

NC 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 Cisco Prime Network Registrar IPAM Executive system by the system administrator. PERL 5 is NOT supplied with Cisco Prime Network Registrar 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.

NC 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

DNS Listener

The DNS Listener process can dynamically collect information from DNS servers about updates to zones and import this information into Cisco Prime Network Registrar IPAM.

The DNS Listener process imports DNS resource records collected via an incremental zone transfer (IXFR) from a DNS server into Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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

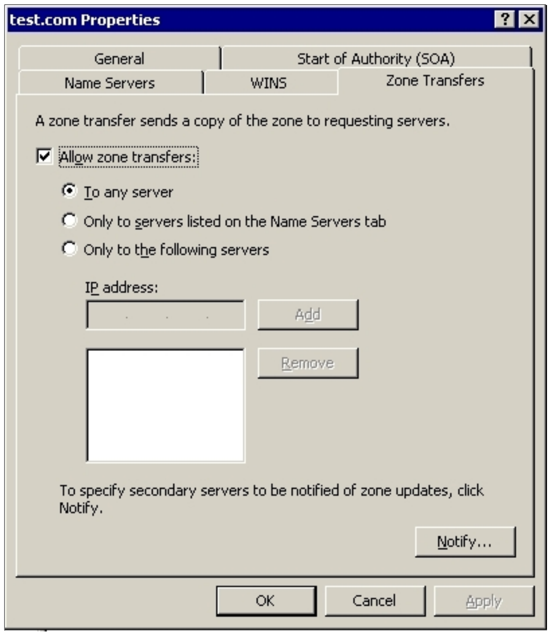
In the BIND configuration file, usually */etc/named.conf*, add the **`also-notify`** option to the zones that you want Cisco Prime Network Registrar IPAM to stay synchronized with. For example, the **`also-notify`** option in this file would cause bind to send notify messages to the address 192.168.0.1 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 Cisco Prime Network Registrar IPAM and select **Properties** from the menu.



test.com Properties

A zone transfer sends a copy of the zone to requesting servers.

[✓] Allow zone transfers:

[● To any server]

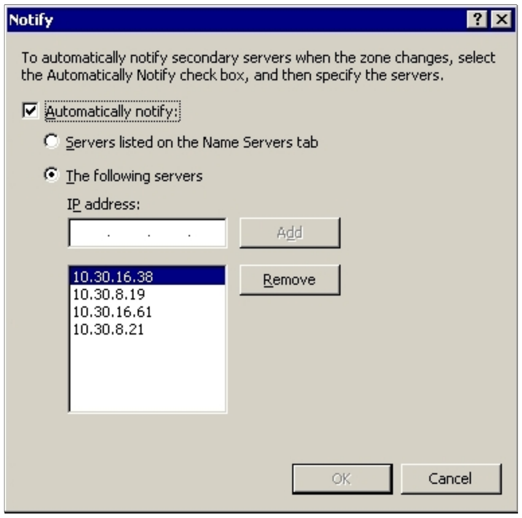
○ Only to servers listed on the Name Servers tab

○ Only to the following servers

IP address:

To specify secondary servers to be notified of zone updates, click Notify.

5. Click on the Zone Transfers tab and then click the **Notify...** button.



Notify

To automatically notify secondary servers when the zone changes, select the Automatically Notify check box, and then specify the servers.

[✓] Automatically notify:

○ Servers listed on the Name Servers tab

[● The following servers]

IP address:

10.30.16.38
10.30.8.19
10.30.16.61
10.30.8.21

6. Select **The following servers** from the radio button group.
7. Add the IP address of the DNS Listener, and any other DNS servers that you want notified when zones are updated.
8. 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 Cisco Prime Network Registrar IPAM. The domain name is taken from the name field of the resource record. If the domain does not exist in Cisco Prime Network Registrar IPAM processing for that record stops. If the domain exists in Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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 Cisco Prime Network Registrar 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.cisco.com. 600 SRV 0 100 400 pdc.sw.cisco.com.` The service specification part would be: `_ldap._tcp.pdc._msdcs` and the domain name part would be: `sw.cisco.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 Cisco Prime Network Registrar IPAM processing of the resource record will stop. If the domain is found in Cisco Prime Network Registrar 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 Cisco Prime Network Registrar IPAM. In this way Cisco Prime Network Registrar 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.

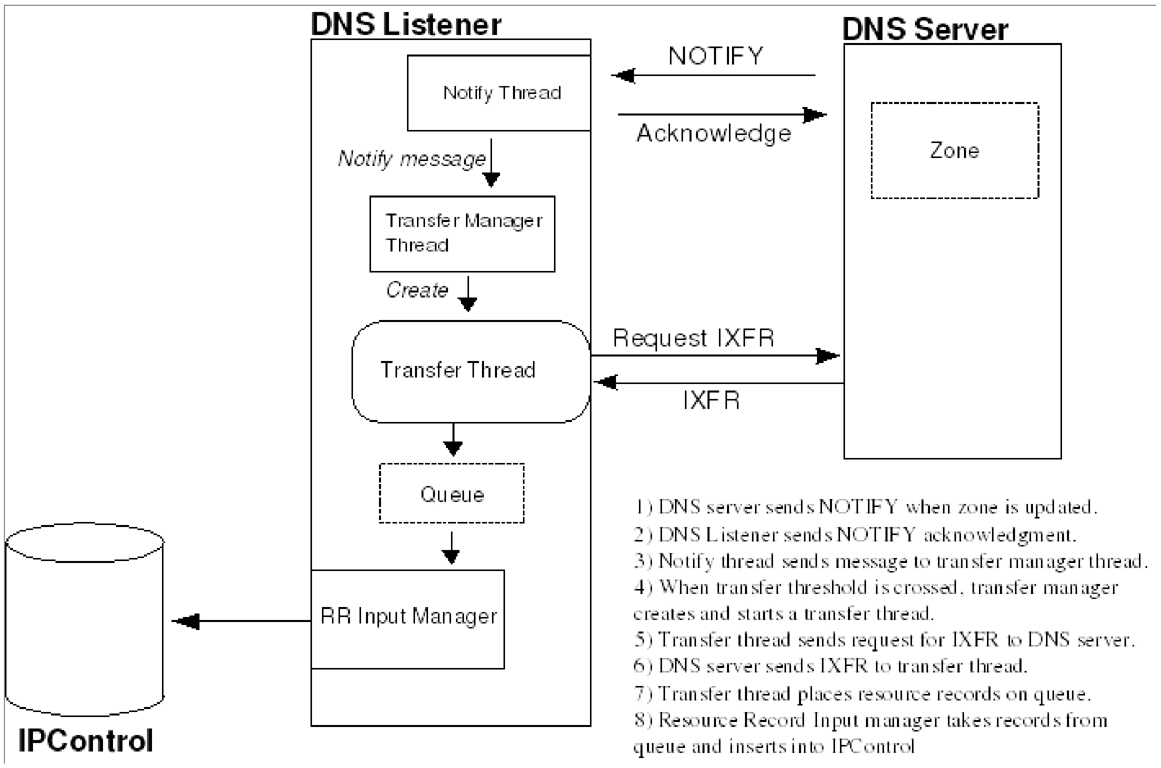

DNS Listener
DNS Server
NOTIFY
Acknowledge
Notify Thread
Zone
Notify message
Transfer Manager Thread
Create
Transfer Thread
Request IXFR
IXFR
Queue
RR Input Manager
IPControl
1) DNS server sends NOTIFY when zone is updated.
2) DNS Listener sends NOTIFY acknowledgment.
3) Notify thread sends message to transfer manager thread.
4) When transfer threshold is crossed, transfer manager creates and starts a transfer thread.
5) Transfer thread sends request for IXFR to DNS server.
6) DNS server sends IXFR to transfer thread.
7) Transfer thread places resource records on queue.
8) Resource Record Input manager takes records from queue and inserts into IPControl

Figure 1: DNS Listeneer

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 Cisco Prime Network Registrar 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.