



SN08 Changes to OSSGate and Its Applications - Highlights

- New T-1 mode introduced for line testing.
- New OSSGate Password Control configuration option.
- Event Messaging support for DQos Service type gateways.
- ITRANS network zone support for interworking with the Session Policy Controller platform and external Virtual Call Admission Control counting.
- Application Layer Gateways (ALG) provisioning and management support through new interface "ALGOSSif".
- Third party gateway profile query functionality from OSS.
- Update to ChangeMG command to change the profile of a provisioned gateway.
- Third party gateway endpoint provisioning commands and support using Batch Provisioning Tool (BPT).
- MG9000 line test interface support using TL1.

0: OSSGate Introduction

OSSGate is an application that provides a machine interface for provisioning components within Succession. The main functionality of OSSGate is to act as a gateway to the Node, Carrier, Trunk, Line, ADSL Provisioning applications and the Trunk Maintenance application. It provides the end user with an alternative to the GUI interface as a method for provisioning succession components; one that allows more automation (less human intervention) than the GUI interface.

OSSGate provides TCP/IP connectivity as an alternative to the asynchronous connections used in the DMS100F. It implements a telnet protocol over a simple TCP/IP socket connection to allow clients to connect to OSSGate. Although it is based on telnet it does not implement a complete set of telnet capabilities. It implements the telnet echo option, which is used to negotiate with telnet client to turn on or off local echo of characters, especially during login time. It does not require the client to support all the telnet options. It does require the client to support line mode. The client or OSS establishes a TCP/IP socket connection to a specific port number (user configurable) on the server running the OSSGate application.

OSSGate supports three modes - XML, CI and TL1. The XML interface is used to send provisioning commands to the Nodes, Carrier, and Trunk applications and maintenance commands to the TMM application. The CI interface is used to send provisioning commands to the Lines Provisioning application. The TL1 interface supported from SN08 is used to send line test commands for MG9000 lines. OSSGate also performs basic syntax checks on XML, TL1 input before sending it to the Nodes, Carrier, ADSL, TMM, Trunk provisioning and LTM line test applications.

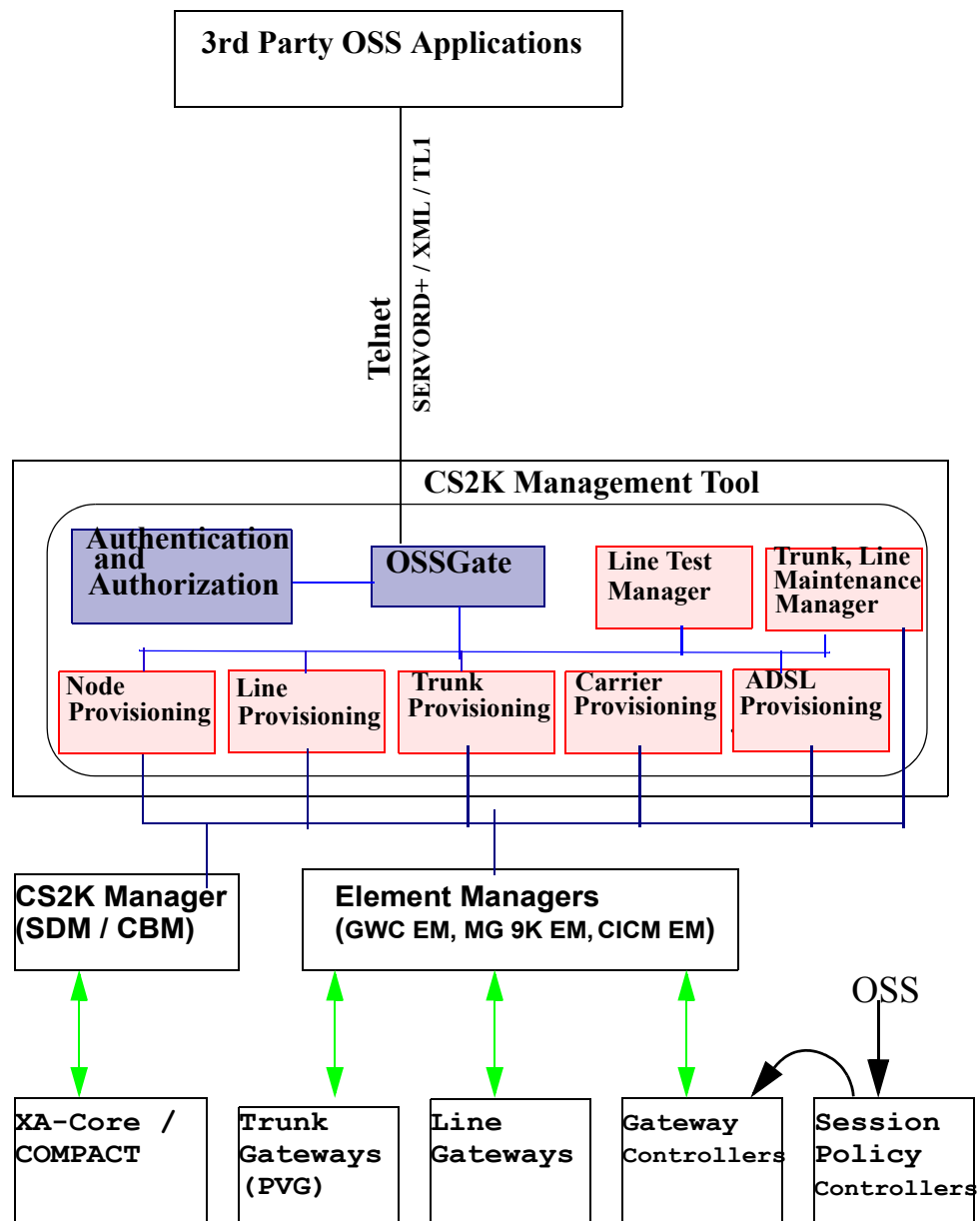


Figure 1 OSSGate interaction with Succession provisioning and Maintenance components in SN08

1: Using OSSGate

The OSSGate server runs as part of the CS2000 Management Tools application. The OSSGate server continuously listens for incoming connection requests. For each connection request, it starts a session after the username and password authentication. Such a session can be used for sending various provisioning commands (for example, SERVORD+ for Line Provisioning commands, XML for Node, Carrier, ADSL and Trunk provisioning). OSSGate can be used by either an OSS or a standard telnet client.

1.1 Supported Provisioning Connections

Following are recommended usage limits apply when each application is being run in isolation (for example, you can have 10 people doing Nodes provisioning as long as zero people are doing Lines, Carriers, and LMM at the same time):

- Node Provisioning (includes add/delete/query of GWs) - Maximum simultaneous users: 10 (only 5 of which can be from the CS2K Management tools GUI)
- Line Provisioning - Maximum simultaneous users: 30
- Carrier Provisioning - Maximum simultaneous users: 4
- LMM - Maximum simultaneous GUI users: 10
- TMM - Maximum simultaneous users: 10
- Trunk Provisioning - Maximum simultaneous users: 5

The maximum number of concurrent OSSGate connections is 45. Following are maximum recommended usage limits for concurrent operations (for example when performing Nodes provisioning, Lines Provisioning, LMM and TMM all at the same time, these limits apply):

- Nodes: 5 users
- Lines: 30 users
- Carriers: 4 users
- LMM: 2 users (GUI)
- TMM: 2 users
- Trunks: 2 users

Note: System response time will increase as the number of concurrent connections increase.

1.2 Connecting to OSSGate via telnet

A system connects to the OSSGate server by initiating a telnet session to the correct host name (or IP address) and port number. user must belong to primary authentication group "succcssn" to login to OSSGate.

```
% telnet <ptm_hostname or IP address> <ossgate_primary_port>
```

Note: The host name or IP address is assigned by the customer network administrator. It is assigned to the server where OSSGate is configured during installation and setup. Refer to your local network administrator for the correct address. The default primary port is 10023 (Standard Telnet Port 23 + 10000).

Example of OSSGate login when OSSGate Password Control option is on¹ (User input is highlighted):

```
$telnet sesmhost 10023
```

```
Trying 47.14.10.45
```

```
Connected to sesmhost.
```

```
Escape character is '^['.
```

```
Enter User Name
```

```
> tom
```

```
Enter Password
```

```
>
```

```
tom logged in on 04/03/25 at 06:10:58.
```

```
*****
**
**          OSS Gateway          **
**
**      This is a PRIVATE Database.      **
**
**      All activity is subject to monitoring.      **
**
**      Any UNAUTHORIZED access or use is PROHIBITED      **
**      and may result in PROSECUTION.      **
**
*****
** WARNING: Different service types require      **
```

```
SN08 Build: Jan 31, 2005 1:36:44 AM
```

```
*****
```

```
>
```

Example of OSSGate login when OSSGate Password Control option is set to default value off² (User input is highlighted):

```
$telnet sesmhost 10023
```

¹ See Section 2.1 which explains about OSSGate Password Control option (default: off)

² See Section 2.1 which explains about OSSGate Password Control option

```
Trying 42.120.94.57...
Connected to sesmhost.
Escape character is '^['.
```

```
Enter username and password
```

```
> maint maint
```

```
maint logged in on 03/11/08 at 08:08:05.
```

```
*****
**
**                                OSS Gateway                                **
**
**          This is a PRIVATE Database.          **
**
**      All activity is subject to monitoring.      **
**
**      Any UNAUTHORIZED access or use is PROHIBITED **
**          and may result in PROSECUTION.          **
**
*****
** WARNING: Different service types require **
** different formats. Consult the OSSGate User's **
** Guide, NE1004-512, for detailed information **
** regarding command formats and usage rules.    **
** Failure to do so can cause a mismatch between **
** XACore and SESM data.                          **
*****
SN08 Build: Jan 31, 2005 1:36:44 AM
*****
>
```

1.3 OSS/Telnet Client Requirements

1.3.1 Non-Secure channel between client and OSSGate

In order for a telnet connection into OSSGate to work, there are some requirements that the telnet client must support. Listed below are the options that the telnet client should support

- Support Line mode. For example, the telnet client should NOT send character by character to OSSGate, instead it should be able to send one line at a time.
- The telnet client should be able to implicitly add a Carriage Return (CR) to any data coming from the OSSGate server
- The telnet client should implement telnet echo option since this will be negotiated by OSSGate to turn on/off during login. Any client that supports these options will work.

1.3.1.1 Putty Configuration

A freeware windows telnet client (putty.exe) which supports the above options can be found at

<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>.

After starting up putty, remember to set the above options by checking the following in the Terminal category:

- Implicit CR in every LF
- Auto wrap mode initially on

Note: In SN08 OSSGate was tested using Putty Release beta 0.53b version on Win2000 machine.

Nortel neither recommends nor provide support for putty.

Unix telnet clients work with OSSGate without need of any explicit options setting.

1.3.2 Setting Secure channel between client and OSSGate using SSH Port Forwarding.

In order to have a secure (i.e., encrypted) channel of communication between OSS/telnet clients and OSSGate server, SSH port forwarding needs to be set up. Refer to the instructions that follow on how to do this for Unix and Windows. Once set up, SSH Port Forwarding establishes a port forwarding session from client to server, wherein all data forwarded are encrypted and hence secure. Regardless of Unix or Windows client, when using SSH port forwarding, OSSGate needs to be configured to set `ssh port_forward=yes`, since this will disallow any non-secure telnet clients to connect to OSSGate. Shown below is example of how to set up port forwarding using openSSH implementation on Unix. The list of SSH clients, both commercial and otherwise for various platforms, is given in section "Other SSH Products."

Unix:

To set up SSH Port Forwarding on the Unix machine, install the SSH software first. After installation, establish a port forwarding session between this machine and the CS2000 management tool server. The following command does that.

```
# ssh -L <local-port>:<remote-host>:<remote-port>  
remote-Host
```

The value for remote-port is typically 10023 (unless OSSGate is configured to listen on a different port). The value for remote-host is the host name of the CS2000 Management Tools server. User can choose any value for

local-port, but higher numbers like 2000 and above are preferred.

The first time this command is run on the user's machine in an attempt to forward data to remote-host, the user will be prompted with information as shown here:

The authenticity of host 'remote-Host (1.2.3.4)' can't be established. RSA key fingerprint is <finger print information>. Are you sure you want to continue connecting (yes/no)?

SSH is verifying with the user whether the host remote-host is a trusted host and whether the user wants to continue connecting to it. The user should enter 'yes'. Next the user will be prompted for a password for the current user logged in. Once the entered password is verified, a successful port forwarding session is established. This means that all new sessions by this user connecting to localhost:local-port will be forwarded to remote-host:remote-port in a secure channel. See the example below:

For example, a user sets up SSH port forwarding on machine A with the following values:

**# ssh -L 2001:CS2000 Management Tools host:10023
CS2000 Management Tools host**

Now, to securely transmit data from machine A to the server (where OSSGate is running), the user needs to open a window logged into machine A and type the following:

telnet localhost 2001

The telnet connection automatically gets secured between machine "A" and the server running the CS2000 Management Tools applications. The OSSGate session will look normal, the user does not see any visible change.

Windows:

The freeware telnet client putty mentioned in the section "Non-Secure channel between client and OSSGate" also supports SSH. In order to secure the connection between the

telnet client and OSSGate server, the user has to configure the SSH port forwarding using the 'putty' software release 0.53B as shown below:

1. Choose Session Category
 - Enter Host name with value of the SUN server that is running the CS2000 Management Tools applications
 - Enter port number as 22
 - Choose protocol as SSH
2. Choose Tunnels Category
 - Under Source Port, give any local port value (say 2001)
 - Under Destination, enter host:port where host is the CS2000 Management Tools host name, and port is the port number where the OSSGate server listens for input. (10023 is the standard port where the OSSGate server listens for a client connection).
 - Choose 'Local' radio button, and click Add.
 - Click on Open.

The user will be prompted to enter a user and password to log in to the CS2000 Management Tools application host/server. Once a valid user and password is entered, this port forwarding session is established.

The user can now establish a secure connection between the client machine and CS2000 Management Tools host as long as the port forwarding session (above) exists, by doing the following.

- Right-click on top left side of the window. From the pull-down menu, choose New Session. A Putty Configuration dialog window opens up.
- Under Session category, enter the string 'localhost' under HostName field. Enter port 2001, since the local port chosen was 2001 in our example (above) in the port field. Choose telnet protocol.
- Choose Terminal options as explained in the section "Putty Configuration". Click on Open.

A secure session with OSSGate server is established as detailed above

Note: Shown above are examples to set up SSH port forwarding to one CS2000 Management Tools host from

a client machine. If a client machine is to have multiple port forwarding sessions, the user must choose different 'local' port numbers. In the examples above, the local port chosen was 2001. If the user wants another port forwarding session to another host, he must choose one.

Please refer chapter 17 for available SSH products and comparison.

1.4 OSSGate commands

The following commands are supported by OSSGate. Note that the following commands are accepted only in Control mode ('?' prompt). To change from input mode ('>' prompt) to Control mode, enter (Ctrl+B) at '>' prompt.

- **login**

Allows user to log in with username and password separated by a space. If a user enters this command when already logged in, OSSGate logs out the previous user. Note that when the user telnets into OSSGate, it automatically prompts for login.

Screen output shown below (User input is highlighted):

```
> ^B
? login
admin logged out.

Enter username and password
> maint maint
maint logged in on 03/11/08 at 08:08:05.

*****
**                                     **
**                               OSS Gateway                               **
**                                     **
**          This is a PRIVATE Database.                                   **
**                                     **
**      All activity is subject to monitoring.                           **
**                                     **
**      Any UNAUTHORIZED access or use is PROHIBITED                    **
**          and may result in PROSECUTION.                               **
**                                     **
*****
** WARNING: Different service types require                             **
** different formats. Consult the OSSGate User's                        **
** Guide, NE1004-512, for detailed information                          **
** regarding command formats and usage rules.                          **
** Failure to do so can cause a mismatch between                        **
** XACore and SESM data.                                                 **
*****

SN08 Build: Jan 31, 2005 1:36:44 AM

*****
>
```

If a login user id or password is invalid or the user does not belong to the primary authentication group “succssn” login will fail and user will get appropriate error response

Enter username and password

> maint maint

User Login Failed, Reason: Authentication failed. Invalid Username / Password / Unauthorized Group.

If three consecutive attempts to login fail, the telnet session will terminate.

- **logout**

Allows the user to log out of OSSGate.

> ^B

? logout

maint logged out.

>

The current or new user can login after a logout using the login command

> ^B

? login

Enter username and password

>

- **security**

Allows the user to know what security service is being used by OSSGate.

Example (User input is highlighted):

> ^B

? security

Security service is JAAS/PAM based

- **clearconv**

Short for “clear conversation”, it allows the user to end the OSSGate session.

> ^B

? clearconv

SESSION TERMINATED.

Connection closed by foreign host.

- **sesm_version** - this communicates and displays the version of CS2000 Management Tools

> ^B

? sesm_version

SN07 Build: Jul 22, 2004 12:26:02 PM

- **mode <OSSGate mode>**

When this command is issued without the <OSSGate mode> parameter, it will display the current mode the OSSGate session is using. The default mode upon login is set to CI. OSSGate supports two modes, the CI mode and the XML mode.

Examples: (User input is highlighted):

Example of querying the current mode

> ^B

? mode

Mode is XML.

Example of switching to CI mode

> ^B

? mode ci

Mode is CI

1.5 Disconnecting from OSSGate

To properly disconnect from OSSGate, the user (or OSS client) should first log out from the session and then close the session.

Example: (User input is highlighted):

> ^B

? logout

user1 logged out.

> ^B

? clearconv

SESSION TERMINATED.

Connection closed by foreign host.

Note: A user can terminate a session without explicitly logging out, since the clearconv command automatically

logs out any logged-in user. The logout command is useful, for example, when a user has finished doing the operations and wants to let another user use the same session, the first user can log out and let another user log in without ending the session.

2: Configuring OSSGate

In SN05 and previous releases users were required to edit properties file to change the OSSGate server configuration. The process is now configured differently. Users can now use a command line configuration utility to change the certain permitted default OSSGate server properties, if needed.

2.1 Configuration Properties

The following properties can be configured for the OSSGate server:

- **Telnet primary port**

This property specifies the primary port on which OSSGate would listen for input. The default port is 10023, which can be overridden by changing this property value in the properties file. The maximum value for this port is 65535. If the user is to override this number, it's advisable to choose a high number, since many standard servers like ftp (21), telnet (23) use lower range numbers.

- **Telnet secondary port**

This property specifies the secondary port on which OSSGate would listen for input if OSSGate failed to bind to the port specified in `oss-gate.telnet.primary_port` property. The default value for this port is 11023, which can be overridden by changing this property value in the properties file. The maximum value for this port is 65535.

Note: Check if the port numbers chosen is not already in use by some other standard server - look in the `/etc/services` file and ensure that the port number is not already used by some other application. If a port is chosen which another application **is** already assigned, an failure will occur on one of the two applications attempting to use the port.

- **Telnet session timeout**

This property specifies the time (in seconds) OSSGate will wait before timing out if there is no input on a session. The default timeout value is 600 seconds (10 minutes). The valid range for this property is a positive number less than 4294967296. (4 bytes)

- **SSH port forwarding**

This property, when set to “y” indicates that the connection between OSS and OSSGate is secured using SSH port forwarding. Hence, OSSGate will reject insecure connections coming directly to it from remote hosts. When the value is set to “n” OSSGate will accept connections from remote clients over a non-secure channel.

- **OSSGate Password Control**

This property, has default value of off. When this option is set to 'on', and the user telnets into OSSGate, the user name is prompted first and then on a second line, the password. The password entered is not echoed to screen. This is more secure option and is recommended to be used. When this option value is set to off (default) the user name and password are prompted in the same line and the password entered is displayed on the screen. Note that changing this option value to on will impact OSS software and they should be ready to handle the change in OSSGate login prompts.

2.1.1 Pre-requisite to reset the properties

- If OSSGate's Telnet Primary port or Telnet Secondary port information is changed, then in order for the changes to be reflected, the CS2000 management tool applications should be stopped (section 2.1.2), the properties value changed (section 2.1.3) and then CS200 management tool applications should be started (section 2.1.4).
- If any other property of OSSGate is changed, then just changing the property values will cause the values to be reloaded (section 2.1.3)

2.1.2 Stopping CS2K management tool server

Warning: The operation detailed below will stop all the CS2000 management tool applications running on the server. It is recommended that the procedure below is during a maintenance window if any of the above properties need to be reset.

Login as user **root** to the CS2K management tool server.

```
#servstop SESMSERVICE
```

This will force stop all the applications on this server, including the proxy agent. The server stop can be verified by using the status option. The status

check must show application as “NOTRUNNING” similar to details shown below.

```
#ptmctl status
```

```
SESM STATUS -----
```

COMPONENT	STATUS
-----	-----
Proxy Agent	NOT RUNNING
RMI Registry	NOT RUNNING
Snmpfactory	NOT RUNNING
MI2 Server	NOT RUNNING

Current number of SESM processes running: 0 (of 4)

SESM APPLICATION STATUS: No Applications are ready

2.1.3 Changing OSSGate parameters

Only user **root** can change the configuration parameters.

```
#configure
```

SESM configuration

- 1 - SESM common configuration (IP addresses, Market, CM CLI)
- 2 - SESM database tools
- 3 - SESM related applications configuration (MG9K, LMM, CICM)
- 4 - SESM provisioning configuration
- 5 - SESM logging configuration (syslog, sesm debug log)
- 6 - view sesm configuration settings
- 7 - SESM refresh properties

X - exit

select - 4

SESM Provisioning Configuration

- 1 - XML Trunk Provisioning configuration
- 2 - OSSGate Provisioning configuration
- 3 - Office Provisioning Type configuration

X - exit

select - 2

SESM OSSGate Provisioning Configuration

Enter the Primary Port OSSGate will use for telnet connections (default: 10023): 11123

Enter the Secondary Port OSSGate will use for telnet connections (default: 11023): 12223

Enter the OSSGate Timeout period, in seconds (default: 600):

Do you wish to have OSSGate Port Forwarding on for SSH [y/n]: n

Do you wish to turn on/off OSSGate Password Control option
(when this option is turned on, OSSGate will not echo
password on screen when logging onto it) [default:off]: on

Primary Port: 11123
Secondary Port: 12223
Timeout: 600
Port Forwarding for SSH: no
Password Control: on

Are these information correct? (y or n): y

Updating the property files with changed values, please wait...

Reloading the changed property values into memory, please wait...

In the above example 11123 was entered for primary port, 12223 was entered for secondary port, a carriage return to pick the default value of 600 for timeout period and “n” for port forwarding on SSH. Once the correct data is entered, user can exit back to command prompt level.

2.1.4 Starting CS2K management tool server

For OSSGate primary and secondary port configuration parameter changes to take effect the CS2K management tool server must be started. To start login as user **root** to the CS2K management tool server.

#servstart SESMSERVICE

Successful startup of the server and all applications, picks up the updated configuration. Users can now telnet to OSSGate using the changed port information.

2.2 Configuring Log levels

CS2000 Management tool server provides ability to run the server and its applications with different log levels. This enables the user to configure the server and applications to generate internal log files during certain situations when errors are detected. The CS2M Management tool fault documentation (Doc no. NN10084911.pdf) details the procedure to set the system to different log levels.

3: User Authorization

3.1 User Groups

Users of OSSGate applications must belong to the primary user group “succsn” and to one or more secondary user groups listed in Table-1.

Table 1: User groups

trkadm	lnadm	mgcadm	mgadm	emsadm
trkrw	lnrw	mgcrw	mgrw	emsrw
trksprov	lnsprov	mgcsprov	mgsprov	emsprov
trkmtc	lnmtc	mgcmtec	mgmtc	emsmtec
trkro	lnro	mgcro	mgro	emsro

A secondary user group consist of a user group domain which defines the range of applications to which a user group applies. The following are valid domains for OSSGate applications:

adm groups have permissions to do everything rw, mtc, sprov and ro can do and more.

rw groups can do everything mtc, sprov and ro can do and more, but can not do adm specific operations.

mtc groups can do everything ro can do and more, but can not do adm, rw or sprov specific operations.

sprov groups can do everything ro can do and more, but can not do adm, rw or mtc specific operations.

ro groups have the least permissions. They can only do read only operations.

The CS2000 Management Tools Security and Administration Guide (NN10172611) lists the various steps to create users, assign users to specific group’s user group domains etc. The mapping of application specific operations to user groups is listed in the application specific chapters discussed in this document. If the user does not have sufficient permissions to execute the command, an error message “Insufficient Security Privileges to perform this action” is returned as part of the response.

3.2 Security Logs

This log is generated when a user logs into the system, attempts to perform an unauthorized operation, a session expires and when user logs out. The log file is located in the server at `/var/log/securitylog` file on the Succession Server Platform Foundation Software (SSPFS) machine.

3.3 Authorization Audit Logs

Authorization Audit Log: this log is generated when a user is successfully authorized to perform an operation. This log does not imply that the user successfully completes the operation. It only indicates the user was granted permission to perform the operation.

This is an Audit Log and the default location for the logs can be found at: `/var/log/auditlog` directory of the SSPFS machine.

3.4 User account, group Administration

For details on how to set user accounts, assign users to different user groups please refer to the CS2000 Management Tool Security and Administration document (NN10172611)

4: Line Provisioning with OSSGate

OSSGate is an application that provides a machine interface for provisioning components within Succession. One of these interfaces is for Line Provisioning (part of the CS2000 Management Tools application). In order to enter Lines Provisioning commands, the OSSGate session must be running in CI mode. CI mode is the default mode when first connecting to OSSGate.

The SERVORD+ syntax for provisioning Succession lines is very similar to the SERVORD syntax used in the DMS. SERVORD commands that are executed from OSSGate are called SERVORD+ commands. With these commands, the endpoint generally replaces the LEN for Succession lines.

For a complete description of the commands and their syntax, please refer to the following NTP reference:

- The SERVORD Reference Manual (NTP 297-8021-808 for North American and NTP 297-9051-8081 for international)

Some examples are given below for the types of commands that can be entered via OSSGate.

Examples:

>QDN <DN>

>QLEN <DR_LEN_TYPE >

>QLEN <MGName> <TPname>

**>NEW \$<DN> 1FR NORT1 0 0 919 NILLATA 0 <MGName>
<TPname> 3WC \$**

>OUT \$9917577 <MGName> <TPname> BLDN

.

Note: The media gateway name (MGname) and the termination point (TP) name have replaced the line equipment number (LEN) wherever the LEN was used in the SERVORD command. Please note that TP alone may not be unique in the succession call server, but the MG and TP pair will uniquely addresses a subscriber line. This can be seen above in the 'OUT' command.

4.0.1 LEN and Gateway/termination Usage in Commands

For most Succession lines, the gateway and termination point name replace the LEN in the command string.

However, there is one gateway type which supports either gateway/termination names or LENSs in the command.

MG9000 - LENSs or gateway/termination names may be used in SERVORD+ commands with certain limitations.

CICM - LENSs must be used for all SN07 format Centrex IP lines. See sect. 4.2.2 below for more details.

For all other gateway types, use of LENSs in non-query commands is not recommended.

4.0.2 DNH Huntgroup DEL Command Syntax Exception

Another change is the modification in the syntax of the DEL command for DNH, DLH, and MLH members. To delete DNH members, users must enter both DNs, MG Name and TP names of the members. The legacy DEL command required only DNs to be entered for the DNH members.

Examples:

```
>DEL $DNH <DN> <MGname> <TPname> $bldn
>DEL $DLH <MGname> <TPname> <key> $
>DEL $MLH <MGname> <TPname> $
```

4.1 Line Provisioning - Special Case For Lengthy SERVORD Commands

When executing Line Provisioning commands that are greater than 75 characters long via a telnet session to OSSGate, the commands must be entered following the convention described in this paragraph. If a SERVORD+ command is greater than 75 characters, the user should put up to 75 characters on one line, then enter a '+' symbol and a carriage return (new line) after the 75th character. The remaining portion of the command should be entered on the next line. Using this convention communicates to OSSGate that the command is continued on the next line.

In the following example, the single SERVORD command is split over five lines of input. Each of the first four lines ends with a '+' followed by a carriage return, and the prompt is given back to the user after each carriage return.

Example:


```
>EST $DNH 9195200570 1FR LATA1 0 les24.mgcp.net aaln/16 9195200571+  
>les24.mgcp.net aaln/17 9195200572 les24.mgcp.net aaln/18 9195200573+  
>les24.mgcp.net aaln/19 9195200574 les24.mgcp.net aaln/20 9195200575+  
>les24.mgcp.net aaln/21 9195200576 les24.mgcp.net aaln/22 9195200577+  
>les24.mgcp.net aaln/23 9195200578 les24.mgcp.net aaln/24 $$15
```

Note: The command when entered at OSSGate can be split at any position.

4.2 SERVORD Commands supported

- Table-2 lists the commands used for provisioning lines. Table-3 lists the line services or options that are available. The commands and options are not described or defined in this document. Refer to the SERVORD Reference Manual (NTP 297-8021-808 for North American and NTP 297-9051-8081 for international) for further details.

4.2.1 MG9000 Flow Through Command Support

The NEW, EST, ADD, DEL, OUT, CDN, and CHDN commands update data on the MG9000 and MG9000 Element Manager. The “NEW”, “EST”, and “ADD” commands add the termination point, directory number and indicated line class code. The “DEL”, and “OUT” commands delete the termination point. The “CDN” and “CHDN” commands change the directory number. CLN, CKLN, and CHG commands are also supported. All flow through to the MG9000EM is blocked when the MG9000 is in an Emergency Stand Alone (ESA). Other conditions within the MG9000EM and MG9000 may also block the successful process of OSSGate commands. Please refer to MG9000 product documentation for more specific information regarding ESA and other failure modes, response.

4.2.2 CICM Flow Through Command Support

SERVORD+ commands are used to ensure that the necessary information is provisioned in the CS2K tables and in the CICM EM for CICM lines. This ensures key labelling consistency between the CICM client and the CS2K representation of the line. The current flow through provisioning model for CICM lines does NOT flow all SERVORD commands through to the CICM.

Note: Hunt group commands (eg. EST/ADD/DEL) do not provision CICM correctly. The use of hunting features on CICM will require separate provisioning via the CICM EM

GUI. The user must manually ensure that the CICM client key labels are updated via the CICM Element Managers. MADN also requires separate provisioning via CICM EM GUI.

Line provisioning commands sent to CICM gateway as part of flow through provisioning include: ADO, CDN, CHF, CHL, DEO, NEW, NEWACD, OUT. CICM client specific SERVORD+ commands (NEW, EST, OUT, etc.) may contain USERID and PASSWD option. The USERID and PASSWD options are used to specify CICM client information in the various SERVORD+ commands which create or modify lines. Data fill interaction include:

1. Key feature addition overwrites any existing key feature on the CICM terminal. No SERVORD equivalent data fill rules are applied to CICM data fill. CS2K SERVORD checking still applies.
2. Assignment of multiple features to a single key is not allowed (e.g., 4 rag 4 3WC). Duplicate feature assignments to key 1 only is allowed (e.g., 1 USERID 1234 \$1 RAG).
3. CICM specific key features USERID and PASSWD must be assigned to key 1. Duplicate feature assignments to key 1 only is allowed (e.g., 1 USERID 1234 \$1 RAG).
4. The USERID feature cannot be changed using the CHF command. Use DEO and ADO commands instead. Note that the password is also reset when the ADO command is executed.
5. The PASSWD feature can be added, changed or deleted at anytime providing that the USERID feature is provisioned.
6. USERID and PASSWD features will never be sent to the CS2K SERVORD interface.
7. CICM and CICM EM applications must be running, otherwise the SERVORD+ command will fail. Commands are not saved and executed at a later date.

4.2.3 CICM Line provisioning examples.

```
1. NEW $8906917 M5216 CSLINES 0 0 125 1 Y CICM 142 2 00 01 3 3WC 4 ACB 1
   $$
   RESULT: features 3WC, ACB, added to CICM LEN
```

2. NEW \$8906917 M5216 CSLINES 0 0 125 1 Y CICM 142 2 00 01 3 3WC 4 ACB 1
\$1 USERID 9999 PROFILE SRV \$1 PASSWD 1234 \$
RESULT: features 3WC, ACB, USERID, PASSWD added to CICM LEN
3. ADO \$CICM 142 2 00 01 5 3WC 6 RAG \$
Result: 3WC added to key 3 and RAG added to key 6.
4. DEO \$ CICM 142 2 00 01 3 3WC 6 RAG \$
Result: 3WC removed from key 3 and RAG removed from key 6.
5. CHF \$CICM 142 2 00 01 5 3WC 6 RAG 7 ACBAMA \$
RESULT: features 3WC and RAG added to keys 5 and 6.
6. OUT \$1278701234 CICM 142 2 00 01 BLDN
RESULT: All features and USERID and PASSWD options deleted from CICM

4.2.4 CICM Line provisioning Responses.

Table-1 lists the new SERVORD command response messages that may be presented to the OSSGate user to indicate a CICM-specific command failure or problem.

Table 1 CICM Line provisioning - response messages.

Response Message	Cause
USERID option change is not supported. Use DEO and ADO commands.	Entered command (e.g., CHF) cannot be used to change the USERID option data.
{0} restricted to key 1. Example: USERID restricted to key 1.	The option string (represented by {0}) can only be added to key 1.
Multiple {0} options are not allowed. Example: Multiple USERID options are not allowed.	The option string (represented by {0}) can only be added to a single key.
Multiple options assigned to keys {0}. Example: Multiple options assigned to keys 3 and 4.	The key numbers (represented by {0}) can only be associated with a single option.
Multiple options assigned to key {0}. Example Multiple options assigned to key 3.	The key number (represented by {0}) can only be associated with a single option.
USERID option syntax is invalid.	The entered USERID option is not compatible with the syntax
PASSWD option syntax is invalid.	The entered PASSWD option is not compatible with the syntax

Table 1 CICM Line provisioning - response messages.

Response Message	Cause
Addon {0} is not valid for LCC {1}. Example: Addon M622 is not valid for LCC M5216.	A valid addon was entered but is not compatible with the indicated LCC.
Addon {0} data is invalid. Quantity must be 0 - {1}. Example: Addon M522 data is invalid. Quantity must be 0 - 2.	Addons can be added in groups. The quantity added is dependant on the addon type.
{0} command with LCC = {1}; {2} is required. Example: NEWACD command with LCC = M5678; M5216 is required.	The SERVORD command ({0}) and LCC ({1}) are not valid for the supported LCC.
NEWACD agent type must be SUPERVISOR or AGENT.	Self explanatory.
Unable to locate ACD Group information for agent.	The SERVORD command syntax is invalid. The command parser was unable to locate the ACD group field(s).
Unable to locate ACD Group information for supervisor.	The SERVORD command syntax is invalid. The command parser was unable to locate the ACD group field(s).
Failed to define ACD options using template.	Internal application error. Collect details and report to the next level of support.
Unable to locate template information.	The entered ACD template cannot be found in the KSETKEYS table. Check the table and try again.
LCC {0} is not valid. Example: LCC M5678 is not valid.	The entered Line Class Code is not supported by the Proxy for the entered CICM LEN.
Template {0} can not be used with the {1} ACD type. Example: Template NORTEL can not be used with the AGENT ACD type.	Data in table KSETKEYS is not consistent with the entered ACD type.
Template {0} is not compatible with the {1} LCC. Example: Template NORTEL is not compatible with the M5316 LCC.	Data in table KSETKEYS is not compatible with the entered LCC.
Template {0} has another {1} option. Only one {1} is allowed on an ACD.	User is attempting to add duplicate options using templates.
Timeout waiting for CICM EM response.	An HTTPS connection was made with the IIS web server but the EM response exceeded the provisioned timeout threshold.

Table 1 CICM Line provisioning - response messages.

Response Message	Cause
Timeout contacting Primary & Secondary CICM EM server.'	The proxy was unable to contact either the primary or secondary IIS servers. Check the state of the IIS servers or provisioned CICM data on SESM.
End of option area expected. '\$' not found.	SERVORD command syntax is invalid. The option string or command must be terminated by a '\$' character.
Unable to parse the SONUMBER.	Self explanatory. SONumber should be '\$'.
Invalid DN entered.	The entered DN, or combination of SNPA and NXX is not valid.
Invalid LEN entered.	The proxy is unable to parse the entered CICM or other LEN.
The NXX of the DN was not found in table TOFCNAME.	Self explanatory.
Rejected by CICM EM gateway. XML Gateway Message failure.	Fatal error. The request sent to the IIS web server was rejected. Collect details and report to the next level of support.
Rejected by CICM EM. EM terminal data corrupted.	The SERVORD request partially failed on the CICM EM and the proxy was unable to successfully undo the changes made.
Command FAILED. Rejected by CICM EM.	The SERVORD request failed on the CICM EM.

4.3 Line Provisioning Commands and Authorization Level

In addition to users belonging to “succssn” group to login to OSSGate, user need to be in application specific authorization groups to perform specific operations. Each Line provisioning command is associated with one or more authorization groups. In order to execute a command, a user must belong to at least one of the associated authorization groups. The groups associated with each OSSGate Node provisioning operation are specified in the table below.

In Table-2 below, the Legacy column indicates if the command may be used on Legacy Lines via the OSSGate interface. The Succession column indicates if the command may be used on Succession Lines via the OSSGate interface.

Table Legend:

- SN0x - Indicates the release in which the command is first supported
- N/S - Indicates the command is not supported
- N/A - Indicates the command is not applicable in the given column
- No termination point info means that in output from query commands, the data will show LEN (logical group) info, but not the corresponding termination point name
- ISDN - Indicates the command is only supported for ISDN phone sets
- MMP - Indicates the command is supported in International markets only

Table 2: Line Provisioning Commands

Number	Command	Legacy	Succession	Authorization Group	Note
1	ABNN	SN05	SN03	Insprov	
2	ADA	SN05	SN05	Insprov	
3	ADD	SN05	SN03	Insprov	
4	ADDPH	SN06	N/A	Insprov	ISDN
5	ADO	SN05	SN03	Insprov	
6	BULK	N/A	N/A	Insprov	
7	CAPSORD	SN06	N/A	Insprov	ISDN
8	CBLKDN	SN06	SN06	Insprov	
9	CDN	SN05	SN05	Insprov	
10	CHDN	SN05	SN03	Insprov	
11	CHAPH	SN06	N/A	Insprov	ISDN
12	CHF	SN05	SN03	Insprov	
13	CHG	SN05	N/S	Insprov	
14	CHL	SN05	SN05	Insprov	
15	CICP	SN05	SN05	Insprov	
16	CISG	SN05	N/A	Insprov	
17	CKLN	SN05	SN05	Insprov	
18	CLN	SN05	SN05	Insprov	
19	CLTG	SN05	SN05	Insprov	
20	COPYSET	SN05	N/S	Insprov	
21	CTP	N/A	SN05	Insprov	
22	CKTP	N/A	SN05	Insprov	

Table 2: Line Provisioning Commands

Number	Command	Legacy	Succession	Authorization Group	Note
23	DBNN	SN05	SN03	Insprov	
24	DEA	SN05	SN05	Insprov	
25	DEL	SN05	SN03	Insprov	
26	DELCF	N/S	N/S	Insprov	MMP
27	DELPH	SN06	N/A	Insprov	ISDN
28	DEO	SN05	SN03	Insprov	
29	DSP	SN05	N/S	Insprov	
30	ECHO	N/S	N/A	Inro	
31	EST	SN05	SN03	Insprov	See restrictions
32	EXBADD	N/S	N/S	Insprov	
33	EXBADO	N/S	N/S	Insprov	
34	EXBCHG	N/S	N/S	Insprov	
35	EXBDEL	N/S	N/S	Insprov	
36	EXBDELG	N/S	N/S	Insprov	
37	EXBDELM	N/S	N/S	Insprov	
38	EXBDEO	N/S	N/S	Insprov	
39	EXBEST	N/S	N/S	Insprov	
40	MQDN	N/S	N/S	Insprov	Wireless only
41	NEW	SN05	SN03	Insprov	
42	NEWACD	SN05	SN06	Insprov	
43	NEWDN	SN05	SN05	Insprov	
44	OUT	SN05	SN03	Insprov	
45	OUTDN	SN05	SN05	Insprov	
46	PLP	SN05	SN05	Insprov	
47	QX75	SN06	N/A	Inro	ISDN
48	QBB	SN06	N/A	Inro	ISDN
49	QBERT	SN05	N/S	Inro	
50	QCM	SN05	SN06	Inro	
51	QCOUNTS	SN05	SN05	Inro	
52	QCPUGNO	SN05	SN06	Inro	
53	QCUST	SN06	SN06	Inadm	
54	QDCH	SN06	SN06	Inro	ISDN

Table 2: Line Provisioning Commands

Number	Command	Legacy	Succession	Authorization Group	Note
55	QDN	SN05	SN03	Inro	
56	QDNA	SN05	SN05	Inro	
57	QDNSU	SN06	SN06	Inadm	
58	QDNWRK	SN06	SN06	Inadm	
59	QGRP	SN05	SN06	Inro	
60	QHA	SN06	SN06	Inadm	
61	QHASU	SN06	SN06	Inadm	
62	QHLR	N/S	N/S	Inro	Wireless only.
63	QHU	SN06	SN06	Inadm	
64	QIT	SN06	N/A	Inro	ISDN
65	QLEN	SN03	SN03	Inro	
66	QLENWRK	SN06	SN06	Inadm	
67	QLOAD	SN06	SN06	Inadm	
68	QLRN	SN05	SN05	Inro	
69	QLT	SN05	N/A	Inro	
70	QMADN	SN06	SN06	Inadm	
71	QMODEL	SN05	SN05	Inro	
72	QMSB	SN05	SN05	Inro	
73	QNCOS	SN06	SN06	Inadm	
74	QPDN	SN06	SN06	Inadm	
75	QPHF	SN05	N/A	Inro	
76	QPRI0	SN05	SN05	Inro	
77	QSCONN	SN06	N/A	Inro	ISDN
78	QSCUGNO	SN05	SN06	Inro	
79	QSIMR	SN05	SN05	Inro	
80	QSL	SN05	SN06	Inro	
81	QTOPSPOS	SN05	SN05	Inro	
82	QTP	N/A	SN04	Inro	
83	QWUCR	SN05	SN05	Inro	
84	RES	SN05	SN05	Insprov	
85	RESGRP	SN05	SN05	Insprov	
86	SDNA	SN05	SN05	Insprov	

Table 2: Line Provisioning Commands

Number	Command	Legacy	Succession	Authorization Group	Note
87	SETPH	SN06	N/A	Insprov	ISDN
88	SLT	SN04	N/A	Insprov	
89	STOPECHO	N/S	N/A	Insprov	
90	SUS	SN05	SN05	Insprov	
91	SUSGRP	SN05	SN05	Insprov	
92	SWAP	N/S	N/S	Insprov	
93	SWLT	SN06	N/A	Insprov	ISDN

4.4 Line Provisioning Options

In Table-3 below, the Legacy column indicates if the option may be used on Legacy Lines via the OSSGate interface. The Succession column indicates if the option may be used on Succession Lines via the OSSGate interface. Not all options are valid for all line class codes. For more details, refer to the appropriate SERVORD NTPs.

Table Legend

- SN0x - Indicates the first release in which the option is supported
- N/S - Indicates the option is not supported
- N/A - Indicates the option is not applicable in the given column
- ISDN - Indicates the option is only supported for ISDN phone sets
- MMP - Indicates the option is supported in International markets only

NOTE: Any options not listed in the above table are considered not supported via the OSSGate Line Provisioning interface.

Table 3: Line Provisioning Option

Number	Option	Legacy	Succession	Notes
1	1MMS	SN05	N/S	
2	3WC	SN05	SN05	

Number	Option	Legacy	Succession	Notes
3	3WCPUB	SN05	SN05	
4	AAB	SN05	SN05	
5	AAK	SN05	SN05	
6	ABR	SN05	SN05	
7	ACB	SN05	SN05	
8	ACCB	SN05	SN05	
9	ACD	SN05	SN05	
10	ACDNR	SN05	SN05	
11	ACOU	SN05	N/A	ISDN
12	ACR	SN05	N/A	ISDN
13	ACRJ	SN05	SN05	
14	ADSI	SN05	SN05	
15	ADSL	SN06	N/S	
16	AEMK	SN05	SN05	
17	AFC	SN05	N/A	ISDN
18	AGA	SN06	N/A	ISDN (Using SLT command)
19	AIN	SN05	SN05	
20	AINDENY	SN06	SN06	
21	AINDN	SN05	SN05	
22	AINMWT	SN05	SN05	
23	AIOD	SN05	SN05	
24	ALI	SN06	SN06	
25	AMATEST	SN05	SN05	
26	AMSG	SN05	SN05	
27	AMSGDENY	SN05	SN05	
28	AOC	SN06	SN06	MMP
29	APS	SN05	SN05	
30	AR	SN05	SN05	
31	ARDDN	SN05	SN05	

Number	Option	Legacy	Succession	Notes
32	ARLNA	SN05	SN05	
33	ARR	SN06	N/A	ISDN
34	ASL	SN05	SN05 ^a	
35	ASP	SN05	SN05	
36	ATC	SN05	SN05	
37	AUD	SN05	SN05	
38	AUL	SN05	SN05	
39	AUTODISP	SN05	SN05	
40	AUTOHF	N/S	N/S	ISDN MFT sets
41	AVT	N/S	N/S	
42	BBGI	SN06	N/A	ISDN
43	BC	SN05	SN05	
44	BCLID	SN05	N/S ^b	
45	BLF	SN05	SN05	
46	BLOCKCDN	SN06	N/A	ISDN
47	BLOCKCGN	SN06	N/A	ISDN
48	BNN	SN05	SN05	See restrictions
49	BRICLID	SN06	SN06	MMP
50	CACH	SN06	N/A	ISDN (using SLT command)
51	CAG	SN05	SN05	
52	CALLOG	SN05	SN05	
53	CARR	SN06	SN06	MMP
54	CARRG	SN06	SN06	MMP
55	CBE	SN05	SN05	
56	CBI	SN05	SN05	
57	CBU	SN05	SN05	
58	CCBS	SN05	SN05	
59	CCNR	SN06	SN06	MMP
60	CCSA	SN05	SN05	

Number	Option	Legacy	Succession	Notes
61	CCV	SN05	SN05	
62	CCW	SN05	SN05	
63	CCWB	SN06	SN06	MMP
64	CD	SN06	SN06	MMP
65	CD0–CD9	SN05	N/S	
66	CDC	SN05	SN05	
67	CDE	SN05	SN05	
68	CDF	SN05	SN05	LCC
69	CDI	SN05	SN05	
70	CDND	SN06	SN06	MMP
71	CDT	SN05	SN05	
72	CDTA	SN06	SN06	MMP
73	CDTO	SN06	SN06	MMP
74	CDU	SN05	SN05	
75	CEPT	SN06	SN06	MMP
76	CFB	SN05	SN05	
77	CFBL	SN05	SN05	
78	CFD	SN05	SN05	
79	CFDA	SN05	SN05	
80	CFDVT	SN05	SN05	
81	CFF	SN05	SN05	
82	CFFA	SN06	SN06	MMP
83	CFFPOVR	SN05	SN05	
84	CFGD	SN05	SN05	
85	CFGDA	SN05	SN05	
86	CFI	SN05	SN05	
87	CFIND	SN05	SN05	
88	CFK	SN05	SN05	
89	CFMDN	SN05	SN05	

Number	Option	Legacy	Succession	Notes
90	CFO	SN05	SN05	
91	CFRA	SN05	SN05	
92	CFS	SN05	SN05	
93	CFTANN	SN05	SN05	
94	CFTB	SN05	SN05	
95	CFTD	SN05	SN05	
96	CFU	SN05	SN05	
97	CFW	SN05	SN05	
98	CFXDNCT	SN06	N/A	ISDN
99	CFXVAL	SN06	N/A	ISDN
100	CFWANN	SN05	SN05	
101	CHD	SN05	SN05	
102	CHG	SN06	N/A	ISDN
103	CIDSDLV	SN05	N/S	ISUP
104	CIDSSUP	SN05	N/S	ISUP
105	CIF	SN05	SN05	
106	CIR	SN05	SN05	
107	CLF	SN05	SN05	
108	CLI	SN05	SN05	
109	CLNT900	SN05	SN05	
110	CLSUP	SN05	SN05	
111	CMCF	SN05	SN05	
112	CMD	SN06	N/A	ISDN
113	CMG	SN05	SN05	
114	CNAB	SN05	SN05	
115	CNAMD	SN05	SN05	
116	CND	SN05	SN05	
117	CNDB	SN05	SN05	
118	CNDBO	SN05	SN05	

Number	Option	Legacy	Succession	Notes
119	CNDDTMF	SN06	SN06	MMP
120	CNF	SN05	SN05	
121	COD	SN05	SN05	
122	COLP	SN06	SN06	MMP
123	COLR	SN06	SN06	MMP
124	COLROVR	SN06	SN06	MMP
125	COT	SN05	SN05	
126	COTAMA	SN05	SN05	
127	CPC	SN06	SN06	MMP
128	CPH	SN05	SN05	
129	CPR	SN05	N/S	
130	CPU	SN05	SN06	
131	CRA	SN06	SN06	MMP
132	CRBL	SN06	N/A	ISDN
133	CRT	SN05	SN05	
134	CRTDENY	SN05	SN05	
135	CSMI	SN05	SN05	
136	CTD	SN05	SN05	
137	CTW	SN05	SN05	
138	CUG	SN05	SN06	
139	CUSD	SN05	SN05	
140	CW	SN06	SN06	MMP
141	CWD	SN05	SN05	
142	CWI	SN05	SN05	
143	CWO	SN05	SN05	
144	CWR	SN05	SN05	
145	CWT	SN05	SN05	
146	CWTACT	SN05	SN05	
147	CWTC	SN05	SN05	

Number	Option	Legacy	Succession	Notes
148	CWX	SN05	SN05	
149	CXR	SN05	SN05	
150	DASK	SN05	SN05	
151	DBC	SN06	N/A	ISDN
152	DCBI	SN05	SN05	
153	DCBX	SN05	SN05	
154	DCC	SN06	N/A	ISDN
155	DCF	SN05	SN05	
156	DCPK	SN05	SN05	
157	DCPU	SN05	SN05	
158	DCPX	SN05	SN05	
159	DDN	SN05	SN05	
160	DDI	SN06	SN06	MMP
161	DELFTERM	SN06	N/A	ISDN (using SLT command)
162	DENY	SN05	SN05	
163	DENYAR	SN05	SN05	
164	DENYCSMI	SN05	SN05	
165	DENYCTFP	SN05	SN05	
166	DENYCWTC	SN05	SN05	
167	DENYISA	SN05	SN05	
168	DENYSRA	SN05	SN05	
169	DENYU3WC	SN05	SN05	
170	DENYWUCR	SN06	SN06	MMP
171	DGT	SN05	SN05	
172	DIN	SN05	SN05	
173	DISCTO	SN05	N/S	
174	DISP	SN05	SN05	
175	DLH	SN05	SN05	
176	DMCT	SN06	SN06	MMP

Number	Option	Legacy	Succession	Notes
177	DNA	SN06	N/A	ISDN (using SETPH, ADDPH, DELPH, and CHAPH commands).
178	DND	SN05	SN05	
179	DNH	SN05	SN05	
180	DOR	SN05	SN05	
181	DP	SN05	N/S	
182	DPCAR	SN06	N/A	ISDN
183	DPR	SN05	N/S	
184	DQS	SN05	SN05	
185	DQT	SN05	SN05	
186	DRCW	SN05	SN05	
187	DRING	SN05	SN05	
188	DROP	SN06	N/A	ISDN
189	DSCWID	SN05	SN05	
190	DTM	SN05	SN05	
191	EBO	SN05	SN05	
192	EBX	SN05	SN05	
193	ECM	SN05	SN05	
194	ECONF	SN06	SN06	MMP
195	ECT	SN06	SN06	MMP
196	ECTPTOP	SN06	SN06	MMP
197	EHLD	SN05	N/A	ISDN
198	EKTS	SN06	N/A	ISDN (using SLT command)
199	ELN	SN05	SN05	
200	EMK	SN05	SN05	
201	EMR	SN06	SN06	MMP
202	EMW	SN05	SN05	
203	ENQ	SN06	SN06	MMP
204	ESL	SN05	SN05	

Number	Option	Legacy	Succession	Notes
205	EWAL	SN05	SN05	
206	EWALI	SN05	SN06	
207	EXT	SN05	SN05	
208	FAA	SN05	SN05	
209	FANI	SN05	SN05	
210	FC	SN06	N/A	ISDN
211	FCTDINT	SN05	SN05	
212	FCTDNTER	SN05	SN05	
213	FCTDNTRA	SN05	SN05	
214	FDN	N/S	N/S	MMP
215	FGA	SN05	SN05	
216	FIG	SN05	SN05	
217	FNO	SN05	SN05	
218	FNT	SN05	SN05	
219	FPS	SN06	SN06	
220	FRO	SN06	SN06	
221	FRS	SN06	SN06	
222	FSR	SN05	SN05	
223	FTRGRP	SN05	SN05	
224	FTRKEYS	SN05	SN05	
225	FTS	SN05	SN05	
226	FXR	SN05	SN05	
227	GIAC	SN05	SN05	
228	GIC	SN05	SN05	
229	GLITE	N/S	N/S	
230	GLTC	SN05	SN05	
231	GND	SN05	SN05	
232	HF	N/S	N/S	ISDN MFT sets.
233	HFMUTE	N/S	N/S	ISDN MFT sets.

Number	Option	Legacy	Succession	Notes
234	HLD	SN05	SN05	
235	HOLD	SN06	SN06	MMP
236	HOT	SN05	SN05	
237	HTL	N/S	N/S	MMP
238	I3WC	SN06	SN06	MMP
239	I6WC	SN06	SN06	MMP
240	ICM	SN05	SN05	
241	ICR	SN06	SN06	MMP
242	ICSDEACT	SN05	SN05	
243	ICT	SN06	SN06	MMP
244	ICWT	SN06	SN06	MMP
245	IECFB	SN05	SN05	
246	IECFD	SN05	SN05	
247	IICB	SN06	SN06	MMP
248	ILB	SN05	SN05	
249	ILDCHNL	N/S	N/A	
250	ILR	SN06	SN06	MMP
251	IMB	SN05	SN05	
252	INSPECT	SN05	SN05	
253	INT	SN05	SN05	
254	INTPIC	SN05	SN05	
255	IRR	SN05	SN05	
256	ISA	SN05	SN05	
257	ISADEACT	SN05	SN05	
258	ISDNAMA	SN06	N/A	ISDN
259	IWUC	SN06	SN06	MMP
260	JOIN	SN05	SN05	
261	KSH	SN05	SN05	
262	KSMOH	SN05	SN05	

Number	Option	Legacy	Succession	Notes
263	LAPB	SN06	N/A	ISDN (using SETPH, ADDPH, DELPH, and CHAPH commands)
264	LAPD	SN06	N/A	ISDN (using SETPH, ADDPH, DELPH, and CHAPH commands)
265	LCDR	SN05	SN05	
266	LDA	SN06	SN06	MMP
267	LDSA	SN05	SN05	
268	LDSO	SN05	SN05	
269	LDSR	SN05	SN05	
270	LDST	SN05	SN05	
271	LDTPSAP	N/S	N/S	
272	LINEPSAP	SN05	SN05	
273	LMOH	SN05	SN05	
274	LNPTST	SN05	SN05	
275	LNR	SN05	SN05	
276	LNRA	SN05	SN05	
277	LOB	SN05	SN05	
278	LOD	SN05	SN05	
279	LOR	SN05	SN05	
280	LPIC	SN05	SN05	
281	LROA	N/S	N/S	MMP
282	LRN	SN06	SN06	MMP
283	LRS	N/S	N/S	MMP
284	LSPAO	SN05	SN05	
285	LSPSO	SN05	SN05	
286	LVM	SN05	SN05	
287	M518	SN05	SN05	
288	M522	SN05	SN05	
289	M536	SN05	SN05	

Number	Option	Legacy	Succession	Notes
290	M622	SN05	SN05	
291	MAN	SN05	N/S	
292	MBK	SN05	SN05	
293	MBSCAMP	SN05	SN05	
294	MCH	SN05	SN05	
295	MCID	SN06	SN06	MMP
296	MDN	SN05	SN05	
297	MDNNAME	SN05	SN05	
298	MEMDISP	SN05	SN05	
299	MLAMP	SN05	SN05	
300	MLH	SN05	SN05	
301	MMI	SN06	SN06	
302	MOG	SN05	SN05	
303	MPB	SN05	SN06	
304	MPH	SN06	N/S	
305	MREL	SN05	SN05	
306	MRF	SN05	SN05	
307	MRFM	SN05	SN05	
308	MSB	SN05	SN05	
309	MSBI	SN05	SN05	
310	MSGDEACT	SN05	SN05	
311	MTR	SN06	SN06	MMP
312	MUTE	N/S	N/S	ISDN MFT sets.
313	MWI	SN06	SN06	MMP
314	MWIDC	SN05	SN05	
315	MWINK	SN06	SN06	
316	MWQRY	SN05	SN05	
317	MWT	SN05	SN05	
318	NAME	SN05	SN05	

Number	Option	Legacy	Succession	Notes
319	NCCW	SN05	SN05	
320	NEMD	SN06	SN06	MMP
321	NDC	SN05	SN05	
322	NDNAP	SN06	N/A	ISDN
323	NFA	SN05	N/S	
324	NGTSRVCE	SN05	SN05	
325	NHT	SN05	N/S	
326	NLT	SN05	SN05	
327	NOCOLL	SN05	SN05	
328	NODNY	N/A	N/A	Sub-option of SDN option.
329	NOH	SN05	SN05	
330	NOCOLPSCR	SN06	SN06	MMP
331	NPGD	SN05	N/S	
332	NRS	SN05	N/S	
333	NSDN	SN06	N/S	
334	NTAIT	SN06	SN06	MMP
335	NUMC	SN06	N/A	ISDN
336	OBS	SN05	SN05	
337	OCB	SN06	SN06	MMP
338	OCFA	SN05	SN05	
339	ODB	SN06	N/A	ISDN
340	OFR	SN06	SN06	
341	OFS	SN05	SN05	
342	OLS	SN05	SN05	
343	ONI	SN05	SN05	
344	OUTWT	SN05	SN05	
345	PBL	SN05	SN05	
346	PCA	SN06	SN06	MMP
347	PCACIDS	SN05	N/S	ISDN

Number	Option	Legacy	Succession	Notes
348	PCI	SN06	SN06	MMP
349	PDO	SN05	SN05	
350	PF	SN05	SN05	
351	PIC	SN05	SN05	
352	PILOT	SN05	SN05	
353	PLP	SN05	SN05	
354	PMC	SN05	SN05	
355	PMD	SN06	SN06	
356	PORT	SN05	SN05	
357	PPH	SN06	SN06	MMP
358	PPL	SN05	SN05	
359	PREMTBL	N/S	N/S	Sub-option of SDN option.
360	PRH	SN05	SN05	
361	PRI	SN05	SN05	
362	PRK	SN05	SN05	
363	PRL	SN05	SN05	
364	PROVCDS	SN06	N/A	ISDN
365	PROVCGS	SN06	N/A	ISDN
366	PROVHLC	SN06	N/A	ISDN
367	PROVLLC	SN06	N/A	ISDN
368	PRV	SN05	SN05	
369	PTP	N/S	N/S	MMP
370	PVC	SN06	N/A	ISDN (using SLT command)
371	QBS	SN05	SN05	
372	QCK	SN05	SN05	
373	QTD	SN05	SN05	
374	RAG	SN05	SN05	
375	RATEAREA	SN05	SN05	

Number	Option	Legacy	Succession	Notes
376	RCD	SN05	SN05	
377	RCHD	SN05	SN05	
378	RCVD	SN05	SN05	
379	REASDSP	SN05	SN05	
380	REATTACH	N/S	N/S	
381	RESL	SN06	SN06	MMP
382	REV	SN06	SN06	MMP
383	RLS	SN06	SN06	
384	RND	SN06	N/A	ISDN
385	RMB	SN05	SN05	
386	RMI	SN05	SN05	
387	RMP	SN05	SN05	
388	RMR	SN05	SN05	
389	RMS	SN05	SN05	
390	RMT	SN05	SN05	
391	RPA	SN05	SN05	
392	RSP	SN05	SN05	
393	RSUS	SN05	SN05	
394	SACB	SN05	SN05	
395	SBLF	SN05	SN05	
396	SC1	SN05	SN05	
397	SC2	SN05	SN05	
398	SC3	SN05	SN05	
399	SCA	SN05	SN05	
400	SCF	SN05	SN05	
401	SCI	SN05	SN05	
402	SCL	SN05	SN05	
403	SCMP	SN05	SN05	
404	SCR	SN06	SN06	MMP

Number	Option	Legacy	Succession	Notes
405	SCRJ	SN05	SN05	
406	SCS	SN05	SN05	
407	SCU	SN05	SN05	
408	SCWID	SN05	SN05	
409	SDN	SN05	SN05	
410	SDS	SN05	SN05	
411	SDSDENY	SN05	SN05	
412	SDY	SN05	SN05	
413	SEC	SN05	SN05	
414	SETMODEL	SN05	SN05	
415	SHU	SN06	SN06	
416	SIMRING	SN05	SN05	
417	SKDISP	N/S	N/S	ISDN MFT set.
418	SL	SN05	SN05	
419	SLBRI	SN06	N/A	ISDN (using SLT command)
420	SLC	SN05	SN05	
421	SLQ	SN05	SN05	
422	SLU	SN05	SN05	
423	SLVP	SN05	SN05	
424	SMDI	SN06	SN06	
425	SMDICND	N/S	N/S	
426	SMDR	SN05	SN05	
427	SND	N/S	N/S	Wireless
428	SNUMC	N/S	N/S	MMP
429	SOFTKEY	N/S	N/S	ISDN MFT sets.
430	SOR	SN05	SN05	
431	SORC	SN05	SN05	
432	SPB	SN05	SN05	
433	SPL	N/S	N/S	MMP

Number	Option	Legacy	Succession	Notes
434	SPM	N/S	N/S	MMP
435	SRA	SN05	SN05	
436	SSAC	SN05	SN05	
437	STRD	SN05	SN05	
438	SUBCOM	SN06	SN06	MMP
439	SUPPRESS	SN05	SN05	
440	SUPPRND	SN06	N/A	ISDN
441	SUPR	SN05	SN05	
442	SUS	SN05	SN05	
443	SVCGRP	SN05	SN05	
444	TBO	SN05	SN05	
445	TCW	SN06	SN06	
446	TDN	SN05	SN05	
447	TDV	SN05	SN05	
448	TERM	SN05	SN05	
449	TERML	SN06	N/A	ISDN (using SLT command)
450	TES	SN05	SN05	
451	TFO	SN05	SN05	
452	TLS	SN05	SN05	
453	TRANSFER	SN06	N/A	ISDN
454	TRMBOPT	SN05	SN05	
455	TSPID	SN06	N/A	ISDN (using SLT command)
456	UCD	SN05	SN05	
457	UCDLG	SN05	SN05	
458	UCDLI	SN05	SN05	
459	UCDSD	SN05	SN05	
460	UUS!	SN06	SN06	MMP
461	VAD	SN06	SN06	

Number	Option	Legacy	Succession	Notes
462	VI	SN06	N/A	ISDN
463	VIS	SN06	SN06	MMP
464	VMEADENY	SN05	SN05	
465	VMEADN	SN05	SN05	
466	VMI	SN06	SN06	MMP
467	VOW	SN06	SN06	MMP
468	VOWDN	SN06	SN06	MMP
469	WC	SN05	SN05	
470	WLN	N/S	N/S	MMP
471	WML	SN05	SN05	
472	WUCR	SN05	SN05	
473	XFER	SN06	N/A	ISDN
474	XLAPLAN	SN05	SN05	

- a. SERVORD+ query commands do not display termination point data for this option.
- b. Manual work-around provisioning procedure exists

Note: Any options not listed in the above table are considered not supported via the OSSGate Line Provisioning interface.

4.5 Restricted Queries

The following commands are restricted Queries. Only one restricted query is allowed to run via OSSGate on the CS2K management tool server at a given time.

For example: If one user issues a QDNWRK query via OSSGate and a second user issues a QLENWRK query via a second OSSGate session before the QDNWRK query has finished processing, the QLENWRK query request will be denied, with a message "The maximum number of concurrent instances of this command has been reached. Please try later".

- QCUST
- QDNSU
- QDNWRK

- QHA
- QHASU
- QHU
- QMADN
- QNCOS
- QLENWRK
- QLOAD
- QPDN

In some instances it may be desirable to stop the processing of a restricted query. To stop the processing of a restricted query the KILL command may be issued either via OSSGate or directly from the Core.

Note: The KILL command should be used with extreme caution. It has the potential to kill all of a certain type of process (e.g. all the QDNWRK commands running on the Core).

The following is a list of the restricted queries and their corresponding process names on the Core with example KILL command syntax.

Table 4: Example Syntax to Stop Restricted Commands

Query	Process Name	Example Syntax to Stop Command
QCUST	QCUST	kill QCUST only <process ID> kill QCUST all
QDNSU	SOQDNSU	kill SOQDNSU only <process ID> kill SOQDNSU all
QDNWRK	SOQDWRK	kill SOQDWRK only <process ID> kill SOQDWRK all
QHA	SOQHA	kill SOQHA only <process ID> kill SOQHA all
QHASU	SOQHASU	kill SOQHASU only <process ID> kill SOQHASU all
QHU	SOQHU	kill SOQHU only <process ID> kill SOQHU all
QLENWRK	SOQLWRK	kill SOQLWRK only <process ID> kill SOQLWRK all
QLOAD	SOQLOAD	kill SOQLOAD only <process ID> kill SOQLOAD all

Table 4: Example Syntax to Stop Restricted Commands

Query	Process Name	Example Syntax to Stop Command
QMADN	MDNQUERY	kill MDNQUERY only <process ID> kill MDNQUERY all
QNCOS	SOQNCOS	kill SOQNCOS only <process ID> kill SOQNCOS all
QPDN	QPDNCI	kill QPDNCI only <process ID> kill QPDNCI all

4.5.1 Stopping a Restricted Query

In some instances it may be desirable to stop the processing of a restricted query. To stop the processing of a restricted query the KILL command may be issued either via OSSGate or directly from the Core.

Note: The KILL command should be used with extreme caution. It has the potential to kill all of a certain type of process (e.g. all the QDNWRK queries running on the Core).

4.5.2 Determining the process name of a command:

At the Core issue the query command to determine the process name of the restricted query:

Syntax: `query alias <command>`

```
>query alias qdnwrk
```

```
SOQDWRK  EC=EF01  MODREF=28BA  PERPROCESS  SOQRYSUB
ORIGINAL
```

```
protected:  address=42CD6180  size=00C0 words
```

```
public size=0000 words extension size=0000 words
```

```
shared:      not allocated
```

```
private:     not allocated
```

```
scratch:     not allocated
```

```
entry: QUERY_DN  offset=0000  increment of CIPROC
```

```
alias: QDNWRK
```

SOQDWRK is the process name of the QDNWRK query.

4.5.3 Determining the process IDs of a process:

At the core issue the query command on a process name to determine the process IDs running.

Syntax: *query process <process name>*

```
>query process SOQDWRK
```

```
496C02C0: 451F 208A SOQDWRK class=BKG slice=2
lock=3 unprot=0 queued on flag
```

451F 208A is the process ID of the SOQDWRK (i.e. the QDNWRK command) process running.

4.5.4 Stopping a process:

To stop a specific process ID for a process name issue the KILL command with the process ID either via OSSGate or the Core.

Syntax: *kill <process name> only #<process ID>*

```
> kill SOQDWRK only #451F #208A
```

```
Process 451F 1068 killed dead.
```

To stop all the process IDs for a process name issue the KILL command with the ALL option either via OSSGate or the Core.

Syntax: *kill <process name> all*

```
>kill SOQDWRK all
```

```
Process 451F 1068 killed dead.
```

4.5.5 List of process names for Restricted commands:

Table 5: Example Syntax to Stop Restricted Commands

Query	Process Name	Example Syntax to Stop Command
QCUST	QCUST	kill QCUST only <process ID> kill QCUST all
QDNSU	SOQDNSU	kill SOQDNSU only <process ID> kill SOQDNSU all
QDNWRK	SOQDWRK	kill SOQDWRK only <process ID> kill SOQDWRK all

Table 5: Example Syntax to Stop Restricted Commands

Query	Process Name	Example Syntax to Stop Command
QHA	SOQHA	kill SOQHA only <process ID> kill SOQHA all
QHASU	SOQHASU	kill SOQHASU only <process ID> kill SOQHASU all
QHU	SOQHU	kill SOQHU only <process ID> kill SOQHU all
QLENWRK	SOQLWRK	kill SOQLWRK only <process ID> kill SOQLWRK all
QLOAD	SOQLOAD	kill SOQLOAD only <process ID> kill SOQLOAD all
QMADN	MDNQUERY	kill MDNQUERY only <process ID> kill MDNQUERY all
QNCOS	SOQNCOS	kill SOQNCOS only <process ID> kill SOQNCOS all
QPDN	QPDNCI	kill QPDNCI only <process ID> kill QPDNCI all

4.6 Limitations and Restrictions

- Any commands not listed in Table-3 are considered not supported via OSSGate
- Use of SERVORD commands on Succession Lines via the DMS CallServer MAP (SERVORD) interface is not supported and may cause data mismatches between CS2K management tool server and the CallServer. This is only supported for MG9000 lines with certain restrictions.
 - Only allowed for MG9000 lines with the following restrictions.
 - No service adaptation for MG9000 line cards. Service only supports default service type for each circuit unless manually changed via MG9000 EM and call server tables.
 - No basic ESA (enhanced ESA only). ESA not available until nightly audit runs (runs nightly to download data to MG9000).
- Any options not listed in Table-4 are considered not supported via OSSGate.

- Setting the DMS CallServer table OFCVAR parameter XLAPLAN_RATEAREA_SERVORD_ENABLED to MANDATORY_PROMPTS is not supported for provisioning via the OSSGate Line Provisioning interface.
- Any commands and services that have not been officially tested by Nortel product verification, are considered unsupported and are not officially sanctioned for use.
- The PDO option can not be added on a DN on treatment. The PDO option can not be added or deleted via the table editor. It can only be added or deleted using SERVORD.
- The CHG command does not support changes of LCC (Line Class Code) on lines assigned to an SAA line card that involve changes from business sets to non-business set LCCs or vice-versa.
- There is a limitation on the number of characters that can be entered in one command. This number is approximately 400 characters. If this limit is exceeded, the command will not work. Some commands that allow multiple member adds or deletes can exceed the 400 character limit.
- There is an engineering limit of 16 MADN members in the same group per LGC or gateway. MADN Engineering Rules (SEB 92-11-001). Table OFCENG Office parm MAX_MADN_MEMBERS_PER_LSG
- When the PDO option is applied to a VDN, the VDN can not be removed using the OUTDN command. The PDO option must first be removed using the SERVORD DEO command. This is consistent with existing PDO behavior on DNs associated with POTS, IBN or KSET lines.
- The command name in an error message may not match the command entered when performing certain SERVORD+ commands from OSSGate. The failure message will reflect the actual XA-Core SERVORD command that was executed. Some Examples are: CKTP will be displayed as CKLN and QTP will be displayed as QLEN. Please note that this discrepancy only occurs in error messages.
- Due to performance reasons, it is recommended that system-wide queries be done during off-peak times when no other provisioning is in progress.

4.6.1 HUNT GROUP Restrictions:

The following HUNT group types are supported when using the EST SERVORD+ command:

- MLH, DLH, MPH, DNH, CPU, CMG, SIMRING

The following HUNT group types are not supported when using the EST SERVORD+ command:

- UA, BNN, PRH (NOTE: BNN members may be added via the ABNN command and deleted via the DBNN command).

4.6.2 Meridian Business Group (MBG) Provisioning Limitations:

Office parameter MAX_MBG_LINES (OFCOPT) controls the number of lines allowed to be added to a MBG. When the maximum number of lines is reached, SERVORD+ commands will be rejected by the XACore. However, the XACore SERVORD will not remove HASU LNINV entries which were created as part of the failed processing. These entries will need to be manually removed prior to deleting the associated gateway via the CS2000 Management Tool, however they pose no problems for further line provisioning actions via OSSGate.

4.6.3 CICM Line Provisioning Limitations:

When executing commands (NEW, ADO, DEO, etc.) to add or delete multiple options from CICM (Centrex IP Client Manager) lines, some options may be rejected by the CM while the overall command itself is accepted and processed successfully. For example, when adding or modifying certain line options (e.g., FXR, CAG, DASK, etc.), incompatibilities with existing options can result in a core response message of the form "<feat> COULD NOT BE ADDED" or "...FAILED TO ASSIGN ONE OR MORE FEATURES..." or "<feat> did not pass updating" or "ATTEMPT TO ASSIGN MULTIPLE APPEARANCE". In this case, the command is not rejected in its entirety on the CM and is therefore not rolled back on the CICM Element Manager. Manual action must be taken at the CICM Element Manager to ensure that the option data accepted by the CM is mirrored on the CICM line.

- Hunt Group commands e.g., ABNN, ADD, CHDN, DEL, EST are not supported.
- Centrex IP LCC's cannot be changed (commands CHG, CKLN, CLN).
- COPYSET command is not supported.
- PF and FTRGRP features are not supported.

5: Provisioning Using XML

The greatest use for OSSGate is automation of provisioning and maintenance commands through an OSS connection. Some commands are sent in XML form and then parsed by OSSGate into the parameters required by the interfacing provisioning and maintenance applications. Note: Line Provisioning does not support an XML interface.

Once a connection is established by the OSS, the mode of OSSGate must be set to XML. This is accomplished by sending a CTRL-B and then sending the text string "mode XML". Once in XML mode, the OSS can begin sending the XML data to OSSGate. OSSGate will forward incoming XML input to a parser, which parses the contents of the XML file and, after performing some validation of the parameters, passes the information to the appropriate provisioning or maintenance application. Currently there are two generic formats for a valid XML input file. Both of these are described below.

5.1 Supported Syntax for XML Provisioning

5.1.1 Rules for valid XML commands are:

- For every open tag there must be a close tag.
- Every attribute value must be quoted, either with single or double quotes, but they must be consistent.
- No attribute name may appear more than once in a single element.
- Elements must be nested (closed) properly.
- Only one root element is allowed.
- XML is case-sensitive.
- Support of XML Schemas (rather than DTDs) and XML Name Space is added.

5.1.2 Supported XML file format

5.1.2.1 XML Command Messages

The following changes have been made to the existing Node, Carrier, and V5.2 Carrier Provisioning XML command messages for all existing commands:

- The following XML tag is no longer supported.
 - MethodName
- The following XML tag has been added since SN05
 - the operation name - e.g., when the operation is delete Carrier, a delete Carrier tag exists (where the MethodName tag previously existed).
- standalone="yes" is removed from the first line of the command message.
- The xml version is added to the contents of the "specific method" tag.

Example format of a command message - generic operation/method:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<CommandList >
  <Command>
    <Interface>EndptGrpProvIf</Interface>
    <Methods>
      <MethodName usn="7">someMethod</MethodName>
      <Parameters>
        <fieldName>somevalue</fieldName>
        <fieldName>somevalue</fieldName>
      </Parameters>
    </Methods>
  </Command>
</CommandList>
```

Example format of a command message - generic operation/method:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList >
  <Command>
    <Interface>EndptGrpProvIf</Interface>
    <Methods>
      <someMethod usn="7" version="1.0">
        <Parameters>
          <fieldName>somevalue</fieldName>
          <fieldName>somevalue</fieldName>
        </Parameters>
      </someMethod>
    </Methods>
  </Command>
</CommandList>
```

Example format of a command message, where deleteCarrier is the example:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CommandList >
  <Command>
    <Interface>EndptGrpProvIf</Interface>
    <Methods>
      <MethodName usn="7">deleteCarrier</MethodName>
      <Parameters>
        <mgName>PVG8</mgName>
        <carrierName>DS3_20.1</carrierName>
      </Parameters>
    </Methods>
  </Command>
</CommandList>
```

Example format of an SN05 and later command message - in this example the deleteCarrier method is used:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList >
  <Command>
    <Interface>EndptGrpProvIf</Interface>
    <Methods>
      <deleteCarrier usn="7" version="1.0">
        <Parameters>
          <mgName>PVG8</mgName>
          <carrierName>DS3_20.1</carrierName>
        </Parameters>
      </deleteCarrier>
    </Methods>
  </Command>
</CommandList>
```

Note: For other commands, substitute the command name for deleteCarrier tag. For example, for the associate MG command, substitute assocMG for deleteCarrier. Make sure to do so for both the open and close tags.

5.1.2.2 XML Response Messages

The following changes have been made to the existing Nodes, Carrier, and V5.2 Carrier Provisioning XML response messages, for all existing commands:

- The following XML tags have been removed:
 - MethodName
 - usn
- The following XML tags have been added:
 - the operation name - e.g., when the operation is deleteCarrier, a deleteCarrier tag exists (where the MethodName tag previously existed). This tag also contains the usn.
 - ReturnData

Also, note the following three additional changes to the response message:

- version and usn number are added to the contents of the “operation name” tag. The version refers to the version of the method or operation - e.g., the deleteCarrier operation may have version 1.0 and 2.0, where version 1.0 takes parms X and Y and version 2.0 takes parms X, Y, and Z
- The RC and MsgText XML tags are moved to a new location in the response message. In SN04, they came before any operation-specific data returned in the response message. In SN05 and later, they come after any operation-specific data returned in the response message. The deleteCarrier operation does not reveal the relocation of these tags/fields - there is no other data returned in the response message. However, in other operations, this “re-ordering” is evident.
- standalone="yes" is removed from the first line of the response message.

Example of general file format for a valid XML response file - generic SN04 operation:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CommandList>
  <Response>
    <Interface>somevalue</Interface>
    <Methods>
      <MethodName>somevalue</MethodName>
      <usn>1</usn>
      <RC>somevalue</RC>
      <MsgTxt>somevalue</MsgTxt>
      <ReturnData>
        <Row>
          <fieldName>somevalue</fieldName>
        </Row>
      </ReturnData>
    </Methods>
  </Response>
</CommandList>
```

Example of general file format for a valid XML response file - generic SN05 and later operation:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Response>
    <Interface>somevalue</Interface>
    <Methods>
      <someMethod usn="7" version="1.0">
        <ReturnData>
          <fieldName>somevalue</fieldName>
          <RC>somevalue</RC>
          <MsgTxt>somevalue</MsgTxt>
        </ReturnData>
      </someMethod>
    </Methods>
  </Response>
</CommandList>
```

Example format of an SN04 **deleteCarrier** response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CommandList>
  <Response>
    <Interface>EndptGrpProvlf</Interface>
    <Methods>
      <MethodName>deleteCarrier</MethodName>
      <usn>7</usn>
      <RC>0</RC>
      <MsgTxt>Delete Carrier operation was successful.</MsgTxt>
    </Methods>
  </Response>
</CommandList>
```

Example format of an SN05 and later **deleteCarrier** response

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Response>
    <Interface>EndptGrpProvlf</Interface>
    <Methods>
      <deleteCarrier usn="7" version="1.0">
        <ReturnData>
          <RC>0</RC>
          <MsgTxt>Delete Carrier operation was successful.</MsgTxt>
        </ReturnData>
      </deleteCarrier>
    </Methods>
  </Response>
</CommandList>
```


6: Nodes Provisioning with OSSGate

Nodes Provisioning is available with OSSGate by providing an interface with the CS2000 Configuration Manager (CS2kCfgMgr) that runs as part of CS2000 Management Tools suite. XML requests are sent to OSSGate, which parses the information and forwards it on to the CS2kCfgMgr. Responses are then put into XML-formatted strings that are returned to the interfacing OSS.

6.1 H.323 Provisioning changes

Prior to the SN07 release, for H.323 GWs the End Point Groups (EPGs) were automatically created and added to the system. The size of the EPGs was fixed, dependent upon configuration (North American systems, 24 is used, International systems, 32). The number of EPGs provisioned was automatically calculated by the system so that the appropriate number of groups were added as dictated by the mandatory “Reserved terminations” field. Pre-SN07 the location of the TIDs allocated was under system control. In SN07, this behavior has been altered. EPGs are no longer automatically added. A separate step of adding the EPGs is required using the “Add carrier” operation. Functionality update in changeMG for capacity, IPaddress, port exist. Functionality updates to support H.323 gateways in addCarrier, queryEndpointGroup, listAllCarriers, disassociateMG, deleteCarrier has been added.

6.2 XML Commands

The Cs2kCfgMgr is accessed by way of the cs2kCfgMgrIf interface. The following operations are available through that interface.

- Add a GWC to the callserver specified by XML element addGWCtoCS (not recommended for use by OSS)
- Delete a GWC from the callserver specified by XML element deleteGWCfrmCS (not recommended for use by OSS)
- Query the list of GWCs on the callserver or query the attributes of a specific GWC specified by XML element queryGWC.
- Associate an MG with a GWC specified by XML element assocMG.
- Disassociate an MG specified by XML element disAssocMG

- Change MG attributes specified by XML element `changeMG`.
- Query the list of MGs associated with a GWC or query the attributes of a specific MG specified by XML element `queryMG`.
- Query the list of Sites populated in Call Server table Site specified by XML element `querySiteInfo`.

6.2.1 Add GWC

The following are the XML elements that are applicable to the add GWC operation. Unless indicated the element is required in the XML document to be valid.

addGWctoCS - specifies the provisioning method. It has two attributes `usn` and `version`. The `usn` is a sequence number used by OSS to associate requests with responses. The `usn` value specified in the request is returned in the response. The `version` attribute specifies the version of the method being used. Currently only 1.0.

Parameters - specifies that the following elements are parameters for the method.

csUIName - delimits callserver name value. This parameter element is optional or not required.

gwcUIName - delimits GWC name value.

profileName - delimits gwc profile name value.

gwcActvIp - delimits the active IP address value of the GWC.

gwcSnmpPort - delimits the SNMP protocol port value to be used. This parameter element is optional, or not required.

msgRouterIp - delimits the CM message router IP address value. This parameter element is optional or not required starting SN08.

msgRouterIpPort - delimits the CM message router IP port value. This parameter element is optional or not required starting SN08.

externalIP - delimits the external IP address value for the GWC. This parameter element is optional or not required.

externalPort - delimits the external IP port value for the GWC. This parameter element is optional or not required.

bearerNetworkName - delimits the bearer network name value to which the GWC is being added. This parameter element is optional or not required.

bearerFabricType - delimits the bearer network fabric type value which the GWC must support. This parameter element is optional or not required.

codecProfileName - delimits the code profile name value which is to be applied to the GWC. This parameter element is optional or not required.

termType - delimits terminal type value to be supported by the GWC. This element may be repeated up to 8 times along with a corresponding exec, the following element.

execLineup - delimits executive routine value that is to be associated with the corresponding terminal type. This element may be repeated up to 8 times along with a corresponding terminal type.

6.2.2 Add GWC Response

The following XML elements are returned in response to an ADD GWC operation:

Response - delimits the response.

Interface - delimits the interface specified in the request cs2kCfgMgrIf.

Methods - delimits the cs2kCfgMgrIf methods in the request.

addGWctoCS - method requested. Has attributes usn and version. The user sequence number provided on the request. The version of the method accessed.

RC - return code. Delimits a value indicating the results of the request. See table 7 below for list of valid values.

MsgTxt - delimits text describing the results of the request.

6.2.3 Delete GWC

The following are the XML elements that are applicable to the delete GWC operation. Unless indicated the element is required in the XML document to be valid.

deleteGWCfrmCS - specifies the provisioning method. It has two attributes usn and version. The usn is used by OSS to associate requests with responses. The usn value specified in the request is returned in the response. The version attribute specifies the version of the method being used. Currently only 1.0.

Parameters - specifies that the following elements are parameters for the method.

csUIName - delimits callserver name value. This parameter element is optional or not required.

gwcUIName - delimits GWC name value.

6.2.4 Delete GWC Response

The following XML elements are returned in response to a Delete GWC operation:

Response - delimits the response.

Interface - delimits the interface specified in the request cs2kCfgMgrIf.

Methods - delimits the cs2kCfgMgrIf methods in the request.

deleteGWCfrmCS - method requested. Has attributes usn and version. The user sequence number provided on the request. The version of the method accessed.

RC - return code. Delimits a value indicating the results of the request. See table 7 below for list of valid values.

MsgTxt - delimits text describing the results of the request.

6.2.5 Query GWC

The following are the XML elements that are applicable to the Query GWC operation.

queryGWC - specifies the provisioning method. It has two attributes **usn** and **version**. The **usn** is used by OSS to associate requests with responses. The **usn** value specified in the request is returned in the response. The **version** attribute specifies the version of the method being used. Currently only 1.0

Parameters - specifies that the following elements are parameters for the method.

Note: Note: a choice between each of the following parameters is required, one or the other may be specified.

csUIName - delimits callserver name value. When this parameter element is specified a list of all the GWC's in the call server is returned.

gwcUIName - delimits GWC name value. When this parameter element is specified the attributes of the GWC specified are returned.

6.2.6 Query a List of All GWCs on a Call Server Response

The following XML elements are returned in response to a Query list of all GWCs on a call server operation:

Response - delimits the response.

Interface - delimits the interface specified in the request **cs2kCfgMgrIf**.

Methods - delimits the **cs2kCfgMgrIf** methods in the request.

queryGWC - method requested. Has attributes **usn** and **version**. The user sequence number provided on the request. The version of the method accessed.

ReturnData - delimits the row of the query.

Row - delimits the query data.

gwcUIList - delimits the GWC name value.

gwcIpList - delimits the GWC IP value.

RC - return code. Delimits a value indicating the results of the request. See table 7 below for list of valid values.

MsgTxt - delimits text describing the results of the request.

6.2.7 Query attributes of a GWC Response

The following XML elements are returned in response to a Query attributes of a GWC operation:

Response - delimits the response.

Interface - delimits the interface specified in the request cs2kCfgMgrIf.

Methods - delimits the cs2kCfgMgrIf methods in the request

queryGWC - method requested. Has attributes usn and version. The user sequence number provided on the request. The version of the method accessed.

ReturnData - delimits the row of the query.

Row - delimits the query data.

gwcUList - delimits the GWC name value.

gwcIpList - delimits the GWC active unit IP address value.

callServerId - delimits the call server name value.

nodeName - delimits the GWC's node name value.

typeList - delimits a GWC capability type value.

typeList values are interpreted as follows:

1. Lines
2. Trunks
3. Audio
4. APG
5. DPT
6. DQoS
7. Small MGs
8. Large MGs

9. Audio MGs
10. APG MG's
11. SIPT
12. VRDN
13. RA
14. BCT
15. IPSec
16. Kerberos
17. V52
18. Conferences
19. Announcements
20. RMGC MG's
21. H.323

xacNodeNumber - delimits the call server node number value for the GWC.

activIpAddress - delimits the active GWC IP address value.

snmp port - delimits the value of the port the GWC uses for receiving snmp messages.

mktTones - delimits the tone value for the market where the GWC is deployed.

termTypes - delimits the callserver terminal type value.

pmExecs - delimits the callserver peripheral module executive value. (termTypes and pmExecs are paired corresponding values).

capacity - delimits the value of the terminal capability of the GWC.

RC - return code. Delimits a value indicating the results of the request. See table 7 below for list of valid values.

MsgTxt - delimits text describing the results of the request.

6.2.8 Associate MG

The following are the XML elements that are applicable to the Associate MG operation. Unless indicated the element is required in the XML document to be valid.

Prior to this release, associate MG for a H.323 gateway automatically provisioned the endpoint groups. In SN07 EPGs are now provisioned separately using the "Add Carrier" command.

assocMG - specifies the provisioning method. It has two attributes **usrn** and **version**. The **usrn** is used by OSS to associate requests with responses. The **usrn** value specified in the request is returned in the response. The **version** attribute specifies the version of the method being used. Currently only 1.0.

Parameters - specifies that the following elements are parameters for the method.

mgUIName - delimits MG name value.

mgProfileName - delimits the MG profile name value to be applied to the MG.

mgIpAddr - delimits the MG IP address value.

mgProtocolType - delimits the MG protocol type. Valid values are:

1. NCS
2. Aspen
3. DSMCC
4. MEGACO
5. MGCP
6. H.323

mgProtocolVersion - delimits the version value of the MG protocol to be used.

mgProtocolPort - delimits the port value to be used by the protocol.

mgSiteName - delimits the site name value to used by a line type MG for its logical group. This parameter element is optional or not required.

gwcUIName - delimits GWC name value. This parameter element is optional or not required. When not specified the callserver selects the most appropriate GWC to associate with the MG.

reservedTerminations - delimits the number of terminations value to be reserved for the MG. This parameter element is optional or not required and cannot exceed the engineered or maximum number of terminations supported by the type of MG being associated as defined by the profile.

Note: a choice between each of the following 3 parameters is required, one may be specified.

pepServerName - delimits policy enforcement point server name value. Only valid for MG that support dynamic quality of service or DQoS.

itransMiddleboxName - delimits the adjacent Middlebox name (either a network address translation server name or a VCAC limited bandwidth link name).

rootMiddleboxName1(..5) - delimits the root Middlebox names. Up to 5 separate entries can be defined per MG, tags must be sequential e.g. if 3 root Middleboxes then 1,2 & 3 must be provided.

6.2.9 Associate MG Response

The following XML elements are returned in response to an Associate MG operation:

Response - delimits the response.

Interface - delimits the interface specified in the request cs2kCfgMgrIf.

Methods - delimits the cs2kCfgMgrIf methods in the request.

assocMG - method requested. Has attributes usn and version. The user sequence number provided on the request. The version of the method accessed.

gwcUIName - delimits the name of the GWC that the MG is associated to.

gwcIpAddr - delimits the value of GWC active unit IP address.

mgNodeName - delimits the MGs node name value.

RC - return code. Delimits a value indicating the results of the request. See table 7 below for list of valid values.

MsgTxt - delimits text describing the results of the request.

6.2.10 Disassociate MG

The following are the XML elements that are applicable to the Disassociate MG operation.

disAssocMG - specifies the provisioning method. It has two attributes usn and version. The usn is used by OSS to associate requests with responses. The usn value specified in the request is returned in the response. The version attribute specifies the version of the method being used. Currently only 1.0.

Parameters - specifies that the following elements are parameters for the method.

mgUIName - delimits MG name value of MG to be disassociated.

6.2.11 Disassociate MG Response

The following XML elements are returned in response to an Disassociate MG operation:

Response - delimits the response.

Interface - delimits the interface specified in the request cs2kCfgMgrIf.

Methods - delimits the cs2kCfgMgrIf methods in the request.

disAssocMG - method requested. Has attributes usn and version. The user sequence number provided on the request. The version of the method accessed.

RC - return code. Delimits a value indicating the results of the request. See table 7 below for list of valid values.

MsgTxt - delimits text describing the results of the request.

6.2.12 Change MG

Change MG for H.323 Gateway in SN08 supports ability to change capacity, the IP address, and the Port. The capacity change is a stand alone operation while the IP address and Port changes can be combined.

The following are the XML elements that are applicable to the Change MG operation.

changeMG - specifies the provisioning method. It has two attributes **usrn** and **version**. The **usrn** is used by OSS to associate requests with responses. The **usrn** value specified in the request is returned in the response. The **version** attribute specifies the version of the method being used. Currently only 1.0.

Parameters - specifies that the following elements are parameters for the method.

mgUIName - delimits MG name value.

Note: Only one parameter may be changed per operation, a choice between each set of the following parameters is required, one or the other may be specified.

- a. **reservedTerminations** - delimits the MG profile name value to be applied to the MG.
- b. **mgIpAddr** - delimits the new MG IP address value of an H.323 MG. Not applicable to any other type of MG.
- c. **mgProtocolPort** - delimits the new protocol port value of an H.323 MG. Not applicable to any other type of MG.

Note 1: The user has the ability to include the reserved termination choice or the IP address and/or Port choice, but, cannot list both choices within a single transaction. Since the IP address and Port tags have been made optional, the user can include either or both within a single transaction. Note that failure to include any of the choice tags will result in a failed operation.

- d. **mgProfileName** - delimits the new profile name to which the existing gateway profile wants to change. Change is permitted only if the "from" GWs profile

is set as compatible in the "to" profiles list. In SN08, the following Nortel Profiles are compatible:

- NUERA_GX
 - NUERA_GX_ASPEN
 - NUERA_GX_MEGACO
- PVG_7K
 - PVG_7K_ASPEN
 - PVG_7K_MEGACO
- PVG15K
 - PVG15K_ASPEN
 - PVG15K_MEGACO
- PVG15K_1000
 - PVG15K_1000_ASPEN
 - PVG15K_1000_MEGACO
- PVG15K_PARTIAL
 - PVG15K_PARTIAL_ASPEN
 - PVG15K_PARTIAL_MEGACO
- PVG_VSP3
 - PVG_VSP3_ASPEN
 - PVG_VSP3_MEGACO
- PVG_APG_VSP3
 - PVG_APG_VSP3_ASPEN
 - PVG_APG_VSP3_MEGACO
- PVG_APG
 - PVG_APG_ASPEN
 - PVG_APG_MEGACO

6.2.13 Change MG Response

The following XML elements are returned in response to an Change MG operation:

Response - delimits the response.

Interface - delimits the interface specified in the request cs2kCfgMgrIf.

Methods - delimits the cs2kCfgMgrIf methods in the request.

changeMG - method requested. Has attributes usn and version. The user sequence number provided on the request. The version of the method accessed.

RC - return code. Delimits a value indicating the results of the request. See table 7 below for list of valid values.

MsgTxt - delimits text describing the results of the request.

6.2.14 Query MG

The following are the XML elements that are applicable to the Query MG operation.

queryMG - specifies the provisioning method. It has two attributes usn and version. The usn is used by OSS to associate requests with responses. The usn value specified in the request is returned in the response. The version attribute specifies the version of the method being used. Currently only 1.0.

Parameters - specifies that the following elements are parameters for the method.

Note: a choice between each of the following parameters is required, one or the other may be specified.

gwcUIName - delimits GWC name value. When this parameter element is specified a list of all the MGs associated with the GWC is returned.

mgUIName - delimits MG name value. When this parameter element is specified the attributes of the MG specified are returned.

6.2.15 Query a List of MGs on a GWC Response

The following XML elements are returned in response to a Query list of all MGs on a GWC operation:

Response - delimits the response

Interface - delimits the interface specified in the request cs2kCfgMgrIf.

Methods - delimits the cs2kCfgMgrIf methods in the request.

queryMG - method requested. Has attributes usn and version. The user sequence number provided on the request. The version of the method accessed.

ReturnData - delimits the row of the query.

Row - delimits the query data.

mgUIList - delimits the MG name value.

mgIpList - delimits the MG IP value.

RC - return code. Delimits a value indicating the results of the request. See table 7 below for list of valid values.

MsgTxt - delimits text describing the results of the request.

6.2.16 Query attributes of an MG Response

The following XML elements are returned in response to a Query attributes of an MG operation:

Response - delimits the response

Interface - delimits the interface specified in the request cs2kCfgMgrIf.

Methods - delimits the cs2kCfgMgrIf methods in the request

queryMG- method requested. Has attributes usn and version. The user sequence number provided on the request. The version of the method accessed.

ReturnData - delimits the row of the query.

Row - delimits the query data.

mgUIList - delimits the MG name value.

mgIpList - delimits the MG IP address value.

callServerId - delimits the call server name value.

svcTypeList - delimits an MG service type value

svcTypeList values are interpreted as follows:

- 0 - Line
- 1 - Trunk
- 2 - Audio
- 3 - APG
- 4 - DQoS
- 5 - ITRANS
- 6 - H.323
- 7 - ITRANS ROAM

nodeName - delimits the MGs node name value.

pepServerName - delimits a policy enforcement point server name when applicable (only when the MG supports this attribute e.g. DQOS service type).

itransMiddleboxName - delimits the adjacent Middlebox name (either a network address translation server name or a VCAC limited bandwidth link name) when applicable (only when the MG supports this attribute e.g. ITRANS service type).

engrEndpoints - delimits the number of reserved endpoints value.

protType - delimits the protocol type value. Values mean:

- 1 - NCS
- 2 - Aspen
- 3 - DSMCC
- 4 - MEGACO
- 5 - MGCP
- 6 - H.323

protVersion - delimits the protocol version value.

protPort - delimits the protocol port value.

profileName - delimits the profile name.

rootMiddleboxName1(..5) - delimits the root Middlebox names up to 5 separate entries can be returned per MG when applicable (only when the MG supports this attribute e.g. ITRANS_ROAM service type).

RC - return code. Delimits a value indicating the results of the request. See table 7 below for list of valid values.

MsgTxt - delimits text describing the results of the request.

Note: The pepServerName, itransMiddleboxName and rootMiddleboxName1..5 tags are only returned to the client when the MG service type is compatible (see specific tags above). Also an empty tag can be returned to the client when there is no entry (this is displayed as <none> on the CS2K Config Manager GUI.). The format used for the empty tag is as follows:

<exampleTag></exampleTag>

6.2.17 Get Site

The following are the XML elements that are applicable to the Get Site operation.

getSiteInfoMG - specifies the provisioning method. It has two attributes usn and version. The usn is used by OSS to associate requests with responses. The usn value specified in the request is returned in the response. The version attribute specifies the version of the method being used. Currently only 1.0.

Parameters - delimits the one parameter.

csUIName - delimits the call server name value.

6.2.18 Get Site Response

The following XML elements are returned in response to a Get list of Site names operation:

Response - delimits the response.

Interface - delimits the interface specified in the request cs2kCfgMgrIf.

Methods - delimits the cs2kCfgMgrIf methods in the request.

getSiteInfo - method requested. Has attributes usn and version. The user sequence number provided on the request. The version of the method accessed.

ReturnData - delimits the row of the query.

Row - delimits the query data.

siteNameList - delimits a site name value read from Site table.

RC - return code. Delimits a value indicating the results of the request. See table 7 below for list of valid values.

MsgTxt - delimits text describing the results of the request.

The following are examples of XML commands and the corresponding response messages for each operation supported for Nodes provisioning. The XML code in these examples is formatted for ease of understanding.

Note: **Adding and deleting gateway controllers should be done from the Call Server 2000 GUI rather than via OSSGate.** The syntax is provide for reference.

Figure 1 An example add GWC request

```

<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
<Command>
<Interface>cs2kCfgMgrIf</Interface>
<Methods>
<addGWCtoCS usn="1" version="1.0">
<Parameters>
<csUIName>COMPACT6</csUIName>
<gwcUIName>GWC-12</gwcUIName>
<profileName>LARGE_LINENA</profileName>
<gwcActvIp>26.3.4.8</gwcActvIp>
<gwcSnmpPort>161</gwcSnmpPort>
<msgRouterIp>47.142.128.144</msgRouterIp>
<msgRouterIpPort>4684</msgRouterIpPort>
<bearerNetworkName>NET_IP</bearerNetworkName>
<bearerFabricType>IP</bearerFabricType>
<codecProfileName>Test</codecProfileName>
<termType>POTS</termType>
<termType>KEYSET</termType>
<execLineup>POTSEX</execLineup><execLineup>KSETEX</execLineup>
</Parameters>
</addGWCtoCS>
</Methods>
</Command>
</CommandList>

```

Note: In SN08 the following parameters from the above XML request are made optional tags.

```

<msgRouterIp>.....</msgRouterIp>
<msgRouterIpPort>.....</msgRouterIpPort>

```

Figure 2 An example of an add GWC response

```

<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
<Response>
<Interface>cs2kCfgMgrIf</Interface>
<Methods>
<addGWCtoCS usn="1" version="1.0">
<RC>0</RC>
<MsgTxt>Add GWC operation was successful</MsgTxt>
</addGWCtoCS>
</Methods>
</Response>
</CommandList>

```

Figure 3 An example of assocMG request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>cs2kCfgMgrIf</Interface>
    <Methods>
      <assocMG usn="1" version="1.0">
        <Parameters>
          <mgUIName>mg1.nortel.net</mgUIName>
          <mgProfileName>ASKEY_LINE_GW_4</mgProfileName>
          <mgIpAddr>47.49.23.43</mgIpAddr>
          <mgProtocolType>5</mgProtocolType>
          <mgProtocolVersion>2.0</mgProtocolVersion>
          <mgProtocolPort>2427</mgProtocolPort>
          <gwcUIName>GWC-0</gwcUIName>
        </Parameters>
      </assocMG>
    </Methods>
  </Command>
</CommandList>
```

Note: The parameters need to be specified in correct sequential order as listed in the request command.

Figure 4 An example of assocMG response:

```
<?xml version="1.0" encoding="UTF-8" ?> <CommandList>
  <Response>
    <Interface>cs2kCfgMgrIf</Interface>
    <Methods><assocMG usn="1" version="1.0">
      <ReturnData>
        <Row>
          <gwcUIName>GWC-0</gwcUIName>
          <gwclpAddr>172.17.40.24</gwclpAddr>
          <mgNodeName>LG 00 0</mgNodeName>
        </Row>
        <RC>0</RC>
        <MsgTxt>The MG was successfully associated with a GWC</MsgTxt>
      </ReturnData>
    </assocMG>
  </Methods>
</Response>
</CommandList>
```

H.323 gateway association:

The process for associating an H.323 MG via XML is very similar to adding any other media gateway. The only difference are the additionalMG profiles. The reserved terminations tag is a mandatory tag for H.323 GWs. The reserved endpoints must be specified as the last parameter tag Example:

```
<reservedTerminations>32</reservedTerminations>.
```

Figure 5 Associate H.323 MG

```

<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>cs2kCfgMgrIf</Interface>
    <Methods>
      <assocMG usn="1" version="1.0">
        <Parameters>
          <mgUIName>H323_M1</mgUIName>
          <mgProfileName>SUCCESSION_1000</mgProfileName>
          <mgIpAddr>10.10.10.43</mgIpAddr>
          <mgProtocolType>6</mgProtocolType>
          <mgProtocolVersion>4.0</mgProtocolVersion>
          <mgProtocolPort>6000</mgProtocolPort>
          <gwcUIName>GWC-0</gwcUIName>
          <reservedTerminations>24</reservedTerminations>
          <itransMiddleboxName>NAT1</itransMiddleboxName>
        </Parameters>
      </assocMG>
    </Methods>
  </Command>
</CommandList>

```

Figure 6 An example of disAssocMG request:

```

<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>cs2kCfgMgrIf</Interface>
    <Methods>
      <disAssocMG usn="1" version="1.0">
        <Parameters>
          <mgUIName>mg1.nortel.net</mgUIName>
        </Parameters>
      </disAssocMG>
    </Methods>
  </Command>
</CommandList>

```

Figure 7 An example of disAssocMG response:

```

<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Response>
    <Interface>cs2kCfgMgrIf</Interface>
    <Methods>
      <disAssocMG usn="1" version="1.0">
        <RC>0</RC>
        <MsgTxt>The MG was successfully disassociated from a GWC</MsgTxt>
      </disAssocMG>
    </Methods>
  </Response>
</CommandList>

```

Note: Before a gateway can be deleted, it is important to verify that it is not in use to avoid taking down active calls. All associated

services (lines, carriers, trunks etc. depending on the type of gateway) must be deleted to ensure that no calls can originate during the gateway deletion process. To delete a H.323 gateway “BSY INB” the corresponding trunk group on the CS2K and delete the associated H.323 carrier. Failure to do this will result in the deletion being denied by the system.

Figure 8 An example of **changeMG** request for capacity

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CommandList >
  <Command>
    <Interface>cs2kCfgMgrIf</Interface>
    <Methods>
      <changeMG usn="1" version="1.0">
        <Parameters>
          <mgUIName>mg1.nortel.net</mgUIName>
          <reservedTerminations>3</reservedTerminations>
        </Parameters>
      </changeMG>
    </Methods>
  </Command>
</CommandList>
```

Figure 9 An example of changeMG request for IP address and/or Port

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList >
  <Command>
    <Interface>cs2kCfgMgrIf</Interface>
    <Methods>
      <changeMG usn="1" version="1.0">
        <Parameters>
          <mgUIName>mg1.nortel.net</mgUIName>
          <mgIpAddr>22.44.66.88</mgIpAddr>
          <mgProtocolPort>88</mgProtocolPort>
        </Parameters>
      </changeMG>
    </Methods>
  </Command>
```

Note: The IP/Port change operation is only supported for H.323 GWs and they must have a NAT associated with them.

Figure 10 An example of changeMG request for Profile change

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
<Command>
<Interface>cs2kCfgMgrIf</Interface>
<Methods>
<changeMG usn="1" version="1.0" >
<Parameters>
<mgUIName>MG1</mgUIName>
<mgProfileName>ARRIS_TOUCHTONE_NN03_4</mgProfileName>
</Parameters>
</changeMG>
</Methods>
</Command>
</CommandList>
```

Figure 11 An example of **changeMG** response

```
<?xml version="1.0" encoding="UTF-8"?> <CommandList>
<Response>
<Interface>cs2kCfgMgrIf</Interface>
<Methods><changeMG usn="1" version="1.0">
<RC>0</RC>
<mgUIName>mg1.nortel.net</mgUIName>
<MsgTxt>mg1.nortel.net Change MG Successful
All Change MG requests have been processed.
</MsgTxt>
</changeMG>
</Methods>
</Response>
</CommandList>
```

Figure 12 An example of **changeMG** request that causes a failure response - a change attempted when reserved terminations exceed the maximum

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CommandList>
  <Command>
    <Interface>cs2kCfgMgrIf</Interface>
    <Methods>
      <changeMG usn="1" version="1.0">
        <Parameters>
          <mgUIName>mg1.nortel.net</mgUIName>
          <reservedTerminations>45</reservedTerminations>
        </Parameters>
      </changeMG>
    </Methods>
  </Command>
</CommandList>
```

Figure 13 An example of **changeMG** response - failure scenario

```
<?xml version="1.0" encoding="UTF-8"?> <CommandList>
  <Response><Interface>cs2kCfgMgrIf</Interface>
    <Methods>
      <changeMG usn="1" version="1.0">
        <RC>50</RC><mgUIName>mg1.nortel.net</mgUIName>
        <MsgTxt> mg1.nortel.net Change MG Failed - Reserved Terminations Exceed
          Maximum. All Change MG requests have been processed.
        </MsgTxt>
      </changeMG>
    </Methods>
  </Response>
</CommandList>
```

Figure 14 An example of **getSiteInfo** request

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CommandList>
  <Command>
    <Interface>cs2kCfgMgrIf</Interface>
    <Methods>
      <getSiteInfo usn="1" version="1.0">
        <Parameters>
          <csUIName>RTPS</csUIName>
        </Parameters>
      </getSiteInfo>
    </Methods>
  </Command>
</CommandList>
```

Figure 15 An example of **getSiteInfo** response

```
<?xml version="1.0"encoding="UTF-8"?> <CommandList>
<Response>
  <Interface>cs2kCfgMgrIf</Interface>
  <Methods><getSiteInfo usn="1" version="1.0">
  <ReturnData>
    <Row>
      <siteNameList>HOST</siteNameList>
      <siteNameList>LG</siteNameList>
      <siteNameList>UAIP</siteNameList>
    </Row>
    <RC>0</RC>
    <MsgTxt>Request successful</MsgTxt>
  </ReturnData>
</getSiteInfo>
</Methods>
</Response>
</CommandList>
```

Figure 16 An example of **query all GWC** on a calls server request:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>cs2kCfgMgrIf</Interface>
    <Methods>
      <queryGWC usn="1" version="1.0">
        <Parameters>
          <csUIName>RTPS</csUIName>
        </Parameters>
      </queryGWC>
    </Methods>
  </Command>
</CommandList>
```

Note: The query param in Query Gateway Controller on a CallServer can be either

<csUIName> - Which will list all GWC's in that CM CLLI space.

<gwcUIName> - Which will list the details about specific GWC.

Figure 17 An example of query all GWC on a call server response:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Response>
    <Interface>cs2kCfgMgrIf</Interface>
    <Methods>
      <queryGWC usn="1" version="1.0">
        <ReturnData>
          <Row><gwcUList>GWC-0</gwcUList>
            <gwclpList>172.17.40.24</gwclpList>
          </Row>
          <Row><gwcUList>GWC-1</gwcUList>
            <gwclpList>172.17.40.28</gwclpList>
          </Row>
          <Row><gwcUList>GWC-2</gwcUList>
            <gwclpList>172.17.40.32</gwclpList>
          </Row>
          <Row><gwcUList>GWC-3</gwcUList>
            <gwclpList>172.17.40.36</gwclpList>
          </Row>
        </ReturnData>
        <RC>0</RC>
        <MsgTxt>Query of all the GWC on a Call Server
          was successful</MsgTxt>
      </queryGWC>
    </Methods>
  </Response>
</CommandList>
```

Figure 18 An example query attributes of specific GWC request:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>cs2kCfgMgrIf</Interface>
    <Methods>
      <queryGWC usn="1" version="1.0">
        <Parameters>
          <gwcUName>GWC-1</gwcUName>
        </Parameters>
      </queryGWC>
    </Methods>
  </Command>
</CommandList>
```

Figure 19 Response to query the attributes of a specific GWC:

```

<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
<Response><Interface>cs2kCfgMgrIf</Interface>
<Methods>
  <queryGWC usn="1" version="1.0">
    <ReturnData>
      <Row>
        <gwcUList>GWC-1</gwcUList>
        <gwclpList>172.17.40.28</gwclpList>
        <callServerId>RTPS</callServerId>
        <nodeName>GWC1</nodeName>
        <typeList>2</typeList>
        <typeList>8</typeList>
        <xacNodeNumber>13</xacNodeNumber>
        <actvIpAddress>172.17.40.28</actvIpAddress>
        <snmpPort>161</snmpPort>
        <mktTones>NORTHAM</mktTones>
        <termTypes>ABTRK</termTypes>
        <termTypes>PRAB</termTypes>
        <termTypes>null</termTypes>
        <termTypes>null</termTypes>
        <termTypes>null</termTypes>
        <termTypes>null</termTypes>
        <termTypes>null</termTypes>
        <pmExecs>DTCEX</pmExecs>
        <pmExecs>UTR250</pmExecs>
        <pmExecs>null</pmExecs>
        <pmExecs>null</pmExecs>
        <pmExecs>null</pmExecs>
        <pmExecs>null</pmExecs>
        <pmExecs>null</pmExecs>
        <pmExecs>null</pmExecs>
        <capacity>4094</capacity>
      </Row><RC>0</RC>
      <MsgTxt>Query of a Single GWC was successful</MsgTxt>
    </ReturnData>
  </queryGWC>
</Methods>
</Response>
</CommandList>

```

Figure 20 An example of query attribute of a specific MG request:

```

<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>cs2kCfgMgrIf</Interface>
    <Methods>
      <queryMG usn="1" version="1.0">
        <Parameters>
          <mgUIName>mg1.nortel.net</mgUIName>
        </Parameters>
      </queryMG>
    </Methods>
  </Command>
</CommandList>

```

[Note: The GWC name in gwcUIName tag need not be provided. However, either gwcUIName or mgUIName tag needs to have a value provided. If mgUIName is empty, all the MGs for that GWC will be returned. (queryMGList command)]

Figure 21 An example of query attributes of a specific MG response:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Response><Interface>cs2kCfgMgrIf</Interface>
    <Methods><queryMG usn="1" version="1.0">
      <ReturnData>
        <Row>
          <mgUIList>mg1.nortel.net</mgUIList>
          <mgIpList>47.49.23.43</mgIpList>
          <callServerId>RTPS</callServerId><gwcUIName>GWC-0</gwcUIName>
          <svcTypeList>0</svcTypeList>
          <svcTypeList>1</svcTypeList>
          <nodeName>LG 00 0</nodeName>
          <itranMiddleboxName></itranMiddleboxName>
          <engrEndPoints>2</engrEndPoints>
          <protType>mcp</protType>
          <protVersion>2.0</protVersion>
          <protPort>2427</protPort>
          <profileName>ASKEY_LINE_GW_4</profileName>
          <maxEngrEndPoints>4</maxEngrEndPoints>
        </Row><RC>0</RC>
      <MsgTxt>Query of a SingleMG was successful</MsgTxt>
    </ReturnData>
  </queryMG>
</Methods>
</Response>
</CommandList>
```

Figure 22 An example of query List of MGs on a GWC request:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>cs2kCfgMgrIf</Interface>
    <Methods>
      <queryMG usn="1" version="1.0">
        <Parameters>
          <gwcUIName>GWC-0</gwcUIName>
        </Parameters>
      </queryMG>
    </Methods>
  </Command>
</CommandList>
```

Figure 23 An example of query List of MGs on a GWC response:

```

<?xml version="1.0" encoding="UTF-8"?>
<CommandList><Response>
  <Interface>cs2kCfgMgrIf</Interface>
  <Methods><queryMG usn="1" version="1.0">
    <ReturnData>
      <Row><mgUIList>mg1.nortel.net</mgUIList><mgIpList>47.49.23.43</mgIpList></Row>
      <Row><mgUIList>pp1.arris.net</mgUIList><mgIpList>10.9.1.201</mgIpList></Row>
      <Row><mgUIList>pp10.arris.net</mgUIList><mgIpList>10.9.1.210</mgIpList></Row>
      <Row><mgUIList>pp2.arris.net</mgUIList><mgIpList>10.9.1.202</mgIpList></Row>
      <Row><mgUIList>pp3.arris.net</mgUIList><mgIpList>10.9.1.203</mgIpList></Row>
      <Row><mgUIList>pp4.arris.net</mgUIList><mgIpList>10.9.1.204</mgIpList></Row>
      <Row><mgUIList>pp5.arris.net</mgUIList><mgIpList>10.9.1.205</mgIpList></Row>
      <Row><mgUIList>pp6.arris.net</mgUIList><mgIpList>10.9.1.206</mgIpList></Row>
      <Row><mgUIList>pp7.arris.net</mgUIList><mgIpList>10.9.1.207</mgIpList></Row>
      <Row><mgUIList>pp8.arris.net</mgUIList><mgIpList>10.9.1.208</mgIpList></Row>
      <Row><mgUIList>pp9.arris.net</mgUIList><mgIpList>10.9.1.209</mgIpList></Row>
      <Row><mgUIList>rtps1124a.nortel.com</mgUIList>
      <mgIpList>10.9.1.50</mgIpList></Row>
    <RC>0</RC>
    <MsgTxt>
      Query of all the Media Gateway's on the Gateway Controller was successful
    </MsgTxt>
  </ReturnData>
</queryMG>
</Methods>
</Response>
</CommandList>

```

Figure 24 An example of **Error Response** format for any operation:

```

<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Response>
    <Interface>cs2kCfgMgrIf</Interface>
    <Methods>
      <queryMG usn="1" version="1.0">
        <ReturnData>
          <RC>2</RC>
          <MsgTxt>Operation failed</MsgTxt>
        </ReturnData>
      <queryMG>
    </Methods>
  </Response>
</CommandList>

```

6.3 Policy Enforcement Point Provisioning for Cable

Configuration management of PacketCable dynamic quality of service (DQoS) elements are Policy Enforcement Points (PEPs) Servers, admission control policies, DQoS network level parameters, GWC to PEP Server connections and MG to PEP Server associations.

The primary purpose for implementing DQoS policy enforcement points (PEPs) is to provide subscriber admission control mechanisms and to minimize abusive quality of service (QoS) usage. Abuse occurs when a subscriber fraudulently uses QoS or when denial-of-service attacks are launched.

Policy enforcement is performed by a PEP in the CMTS. Each PEP is associated with a particular MTA (multimedia terminal adapter) or MG that belongs to a subscriber. Each MTA subscribes to a service level or QoS which is granted per session or call. The class of the session identifies the proper admission control policy or parameters to be applied for a call.

Data is configured by a system administrator to change the desired policy that is enforced by the PEP. A separate policy is defined for each level of service that is to be supported in a PacketCable network. Each policy has a DSCP, differentiated services code point and a flow specification.

At this time a single network wide policy will be supported making admission control the primary reason for this capability. Multiple policies i.e. qualities of service is not currently supported.

6.3.1 OSS Based DQoS Configuration Management

Provisioning of the DQoS capability in the Call Server 2000 (CS2k) is accomplished from the CS2000 Management tool GUI. Initial provisioning of a Cable Network supporting DQoS and its elements PEP's and policies proceeds as follows:

- 1 Identify the policy enforcement point servers by adding PEP Servers to the management system. Note: Ensure that the PEP server has been added using the CS2000 management tool GUI.

Add, delete, change of PEP servers are done from the CS2000 Management Tool GUI. Follow details in GWC Configuration NTP NN10112-511.pdf

- 2 Specify the admission control policy to be enforced.

- 3 Associate all MGs (MTAs) with a PEP Server defined in step 1 above.

Once these above steps are completed the network managed from the CS2K management tool is DQoS capable. It is now possible to do any of the following:

- 1 Association of additional MG's with GWCs and PEP Servers by using assocMG xml command.
- 2 Changing an MG's PEP association.
- 3 Monitoring the level of the number of PEP Server to GWC connections and GWC to PEP Server connections to avoid exceeding limits resulting in failures to associate MGs with GWC's by using the QueryPEPData xml command.
- 4 Querying of the admission control policy by using the QueryDQoSPolicy xml command.
- 5 Auditing PEP Servers specified in the network by using the QueryPEPInfo xml command.
- 6 Auditing MG associations with PEP's by using the queryMG xml command.

Figure 25 An example an PEP server use with associate MG request

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
<Command>
<Interface>cs2kCfgMgrIf</Interface>
<Methods>
<assocMG usn="1" version="1.0" >
<Parameters>
<mgUIName>mg5.nortel.net</mgUIName>
<mgProfileName>ARRIS_TOUCHTONE_NN02_4</mgProfileName>
<mgIpAddr>47.143.34.39</mgIpAddr>
<mgProtocolType>1</mgProtocolType>
<mgProtocolVersion>1.0</mgProtocolVersion>
<mgProtocolPort>2724</mgProtocolPort>
<mgSiteName>LG</mgSiteName>
<gwcUIName>GWC-0</gwcUIName>
<reservedTerminations>4</reservedTerminations>
<pepServerName>pep1</pepServerName>
</Parameters>
</assocMG>
</Methods>
</Command>
</CommandList>
```

Figure 26 An example an PEP server use with associate MG response

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
<Response>
<Interface>cs2kCfgMgrIf</Interface>
<Methods>
<assocMG usn="1" version="1.0">
<ReturnData>
<Row>
<gwcUIName>GWC-0</gwcUIName>
<gwclpAddr>172.17.40.24</gwclpAddr>
<mgNodeName>LG 00 0</mgNodeName>
</Row>
<RC>0</RC>
<MsgTxt>The MG was successfully associated with a GWC</MsgTxt>
</ReturnData>
</assocMG>
</Methods>
</Response>
</CommandList>
```

Figure 27 An example of Query Single MG Request with PEP server

```
<?xmlversion="1.0"encoding="UTF-8"?>
<CommandList>
<Command>
<Interface>cs2kCfgMgrIf</Interface>
<Methods>
<queryMGusn="1"version="1.0">
<Parameters>
<mgUIName>gateway1.nortel.net</mgUIName>
</Parameters>
</queryMG>
</Methods>
</Command>
</CommandList>
```

Figure 28 An example of Query Single MG Response with PEP server

Figure 29 An example of Add Network Zone (LBL type) request:

```

<?xmlversion="1.0"encoding="UTF-8"?>
<CommandList>
<Response>
<Interface>cs2kCfgMgrIf</Interface>
<Methods>
<queryMGusn="1"version="1.0">
<ReturnData>
<Row>
<mgUIList>gateway1.nortel.net</mgUIList>
<mgIpList>47.124.142.86</mgIpList>
<callServerId>COMPACT5</callServerId>
<gwcUIName>GWC-6</gwcUIName>
<svcTypeList>0</svcTypeList>
<svcTypeList>4</svcTypeList>
<nodeName>LG001</nodeName>
<pepServerName>pep1</pepServerName>
<algName>NOT_SET</algName>
<engrEndPoints>4</engrEndPoints>
<protType>ncsprotocol</protType>
<protVersion>1.0</protVersion>
<protPort>2427</protPort>
<profileName>ARRIS_TOUCHTONE_NN01_4</profileName>
<maxEngrEndPoints>4</maxEngrEndPoints>
<frameNumber>not_set</frameNumber>
<unitNumber>not_set</unitNumber>
<frameType>not_set</frameType>
<floorPos>not_set</floorPos>
<rowPos>not_set</rowPos>
<framePos>not_set</framePos>
<unitPos>not_set</unitPos>
</Row>
<RC>0</RC>
<MsgTxt>QueryofaSingleMGwassuccessful</MsgTxt>
</ReturnData>
</queryMG>
</Methods>
</Response>
</CommandList>

```


Figure 30 An example request to change assigned PEP server to a media gateway:

Note: The pep server must exist in the system before the change is attempted. In the example mg5.nortel.net assigned to pep1 is changed to pep2. Both pep1 and pep2 are available before the change.

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
<Command>
<Interface>DQoSOSIf</Interface>
<Methods>
<changeMGGWCEMData usn="135" version="1.0" >
<Parameters>
<MGUName>mg5.nortel.net</MGUName>
<PEPServerName>pep2</PEPServerName>
</Parameters>
</changeMGGWCEMData>
</Methods>
</Command>
</CommandList>
```

Figure 31 An example response to change assigned PEP server to a media gateway:

```
<?xml version='1.0'?>
<CommandList>
<Response>
<Interface>DQoSOSIf</Interface>
<Methods>
<changeMGGWCEMData usn="135" version="1.0">
<ReturnCode value="0" text="Successful result" />
</changeMGGWCEMData>
</Methods>
</Response>
</CommandList>
```

Figure 32 An example to get one PEP Server Data request

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
<Command>
<Interface>DQoSOSIf</Interface>
<Methods>
<getPEPServersData usn="379" version="1.0" >
<Parameters>
<PEPServerName>pep1</PEPServerName>
</Parameters>
</getPEPServersData >
</Methods>
</Command>
</CommandList>
```

Figure 33 An example to get one PEP Server Data response

```
<?xml version='1.0'?>
<CommandList>
<Response>
<Interface>DQoSOSsIf</Interface>
<Methods>
<getPEPServersData usn="379" version="1.0">
<ReturnCode value="0" text="Successful result"/>
<PEPServerName>pep1</PEPServerName>
<Type>dqosmb</Type>
<IPAddress>47.143.34.39</IPAddress>
<MaxConnections>10</MaxConnections>
<ProtocolVersion>DQOS I04</ProtocolVersion>
</getPEPServersData>
</Methods>
</Response>
</CommandList>
```

Figure 34 An example of QueryGWCPEPConn request

```
<?xml version="1.0"?>
<CommandList>
<Command>
<Interface>DQoSOSsIf</Interface>
<Methods>
<getPepNamesbyGWCId usn="1" version="1.0">
<Parameters>
<GWC>GWC-4</GWC>
</Parameters>
</getPepNamesbyGWCId>
</Methods>
</Command>
</CommandList>
```

Figure 35 An example of Query GWC to PEP connections response

```
<?xml version='1.0'?>
<CommandList>
<Response>
<Interface>DQoSOSIf</Interface>
  <Methods>
    <getPepNamesbyGWCIId usn="1" version="1.0">
      <ReturnCode value="0" text="Successful result" />
      <GWC>GWC-6</GWC>
      <PEPServer>pep1</PEPServer></getPepNamesbyGWCIId>
    </Methods>
  </Response>
</CommandList>
```

Figure 36 An example of GetDQoSPoliciesData request

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
<Command>
<Interface>DQoSOSIf</Interface>
<Methods>
<getDQoSPoliciesData usn="379" version="1.0" >
  <Parameters>
    <Policy>All</Policy>
  </Parameters>
</getDQoSPoliciesData>
</Methods>
</Command>
</CommandList>
```

Figure 37 An example of GetDQoSPoliciesData response:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
<Response>
<Interface>DQoSOSIf</Interface>
<Methods>
<getDQoSPoliciesData usn="3" version="1.0">
  <RC>0</RC>
  <MsgTxt>Data Available</MsgTxt>
  <DSCP>101110</DSCP>
  <BucketDepth>200</BucketDepth>
  <BucketRate>150</BucketRate>
  <PeakRate>1000</PeakRate>
  <MinimumUnit>16</MinimumUnit>
  <MaxDatagram>1024</MaxDatagram>
  <ReservedRate>4096</ReservedRate>
  <SlackTerm>200</SlackTerm>
</getDQoSPoliciesData>
</Methods>
</Response>
</CommandList>
```

6.3.2 PEP Provisioning Limitations and restrictions

Certain provisioning rules apply when provisioning PEP servers. These include:

- AssocMG.
 - a. The profile applied to the MG must have a service type of DQoS when a PEP Server is specified.
 - b. The GWC specified must have the DQoS capability when a PEP Server is specified.
 - c. The GWC specified cannot have more than 20 PEP Server connections when this limit is exceeded the command is rejected.
 - d. The PEP Server specified cannot have more than the maximum number of connections specified when it was defined. When this limit is exceeded the command is rejected.
- QueryPEPInfo.
 - a. The specified PEP name must be defined.
- QueryGWCPEPData.
 - a. The specified GWC must be defined.

6.4 Event Messaging support for Cable gateways

To meet compliancy to packetcable lab specification, any event message sent from the call server need to contain the gateway and termination information.

To facilitate the above a new table LNENDPT has been added in the CS2K. As part of adding of gateways this table is datafilled for each termination. Lens are thus pre-provisioned. The pre-provisioned lens are in HASU state when the gateways are provisioned. The state changes to WORKING when services are assigned.

6.4.1 Verifying for HASU status

When in CI mode in OSSGate, the QLEN commands response provides an indication of the line status.

Figure 38 QLEN query and response for a HASU Len

```

> qlen LG 00 0 00 01
-----
TYPE: HARDWARE ASSIGNED SOFTWARE UNASSIGNED
CARDCODE: RDTLSG GND: N PADGRP: PKLNL BNV: NL MNO: N
PM NODE NUMBER : 33
PM TERMINAL NUMBER : 2
-----

```

AssociateMG, DissassociateMG and ChangeMG operations facilitates the management of data in the new table.

6.4.2 Upgrade to SN08

The full support for event messaging occurs only after all elements in the network are upgraded. The CS2K is the last in the upgrade order. Refer to upgrade doc NN10061-461 (Upgrading the Communications Server 2000). On upgrade of CS2M to SN08, node provisioning operations will continue to work with the CS2K in a previous supported release. On upgrade of CS2K, a data upgrade process gets initiated, in order to populate the new table LNENDPT. This data transfer process can take some time depending on the switch data fill size. This takes place on a per GWC basis. Any failure of the upgrade process will send an alarm to the fault management systems. Refer to fault management doc. for alarm details. This upgrade alarm details include:

```

Alias Name: EventMessagingUpgrade
Category: Processing Error
Alarm Time: hh:mm:ss DD-MM-YYYY <TimeZone Info>
Severity: Major
Probable Cause: "Upgrade Failure"
Specific Problem: The Event Messaging Upgrade failed or has been interrupted

```

During the upgrade period

- A minor alarm is raised. This alarm indicates the data specific to a GWC being upgraded. The alarm will clear when the specific GWC has been upgraded.

- Node provisioning and line provisioning operations will fail gracefully with the message as shown below.
- The CS2M Line audit operation from CS2K management tool GUI is disabled during the upgrade period. Running a audit can potentially cause incorrect results to be reported.

Figure 39 An example of upgrade-in-progress response

```
<?xml version="1.0" encoding="UTF-8" ?> <CommandList>
<Response><Interface>cs2kCfgMgrlf</Interface>
<Methods>
<changeMG usn="1" version="1.0">
<RC>20</RC><mgUIName>mg1.nortel.net</mgUIName>
<MsgTxt> mg1.nortel.net Change MG Failed.
Data upgrade to SN08 in progress.Please try again later.</MsgTxt>
</changeMG>
</Methods>
</Response>
</CommandList>
```

6.5 User Authorization for Node Provisioning operations

In addition to users belonging to “succssn” group to login to OSSGate, user need to be in application specific groups to perform specific operations. Each operation is associated with one or more user groups. In order to execute a command, a user must belong to at least one of the associated user groups. The user group’s associated with each OSSGate Node provisioning operation are specified in the table below.

Table 6: Node Provisioning user group

Command	User Group				
	mgcadm	mgciprov	mgcmte	mgcsprov	mgcero
disAssocMg	X	X			
assocMG	X	X			
changeMG	X	X			
querySiteInfo	X	X	X	X	X
queryGWC	X	X	X	X	X
queryMG	X	X	X	X	X
changeMGGWCEDData	X	X			
getPEPServerData	X	X	X	X	X

Table 6: Node Provisioning user group

Command	User Group				
	mgcadm	mgciprov	mgcmte	mgcsprov	mgcro
queryGWCPEPConn	X	X	X	X	X
getDQosPoliciesData	X	X	X	X	X
queryEndpointGroups	X	X	X	X	X

6.6 Return Codes

Table 7 shows the Nodes Provisioning return codes generated in SN06.2

Table 7: Nodes Provisioning Return Codes

Return Code	Meaning
0	Successful operation
1	More data coming
2	Request Rejected due to Resource Limits Reached
3	Application was Commanded to Abort
4	Invalid input from client
5	Failed SERVINV update or could not get node name and number
6	Failed Add to GWCEM
7	Failed to update NetworkView database
8	Failed Rollback of ADD GWC
9	Query GWC Operation Failed to Get Data from GWC EM
10	Query GWC Operation Failed to Read GWC List from the Network View
11	Query GWC Operation Failed to Get Data from XA Core
12	Delete GWC Operation failed GWC Data in Network View could not be read
13	Delete GWC Operation aborted, the GWCEM failed to delete the GWC
14	Delete GWC Operation aborted, GWCEM rejected delete, GWC has associated MG's or Endpoints, invalid operation
15	Delete GWC Operation encountered an error after deleting the GWC from the GWCEM
16	Associate MG Operation Invalid input from client interface
17	Associate MG Operation failed to access DB when reading gateway table

Table 7: Nodes Provisioning Return Codes

Return Code	Meaning
18	Associate MG Operation failed to assign a GWC. GWC with sufficient capacity could not be selected
19	Associate MG Operation failed to read the GWC Data frames Network View
20	Associate MG Operation failed to assign a LGRP node name for the MG
21	Associate MG Operation failed to associate the MG with a GWC
22	Associate MG Operation failed to update the SB and the Network View with data about the new MG
23	Associate MG Operation failed roll back of a transaction in progress that encountered an error
24	Query MG Operation invalid input from client
25	Query MG Operation failed to read the Gateway Data from the GWCEM
26	Disassociate MG Operation invalid input from client
27	Disassociate MG Operation failed to Read Gateway Data
28	Disassociate MG Operation failed to read Gateway Controller Data
29	Disassociate MG Operation aborted, GWCEM failed to delete the MG
30	Disassociate MG Operation aborted, GWCEM rejected delete, MG has provisioned Endpoints, this is an invalid operation
31	Disassociate MG Operation failed to de-assign Lgrp Node Name
32	Disassociate MG Operation encountered an error after deleting the MG from the GWCEM
33	Audit of XACore data used by Cs2kCfgMgr failed to complete successfully
34	Query XACore GWC data used by CS2KCfgMgr failed to complete successfully

Table 7: Nodes Provisioning Return Codes

Return Code	Meaning
35	Query XACore GWC data used by CS2KCfgMgr failed to complete IP not found
36	Query XACore GWC data not found in XAC
37	Query XACore GWC data failed
38	Associate MG Operation MG is already provisioned
39	One of the mandatory parameters required for assigning a MG is not present
40	One of the parameters used when assigning a MG does not have the correct format
41	The query of XaCore table site failed
42	The query of XaCore table site returned an empty list
43	The MG name is not known
44	One of the parameters used does not have the correct format
45	One of the mandatory parameters required is not present
46	Other input errors
47	Change MG Operation failed to access NV when reading gateway table
48	Change MG Operation failed to access GWC-EM when reading GWC data
49	Change MG Operation failed to read the Profile Data from the Network View
50	Change MG Operation the number of reserved terminations exceeds maximum value.
51	Change MG Operation the number of reserved terminations less than 1.
52	Change MG Operation the number of reserved terminations less than
53	Change MG Operation the GWC does not have capacity

Table 7: Nodes Provisioning Return Codes

Return Code	Meaning
54	Change MG Operation failed to update the SB and the NetworkView with data about the new MG
55	Change MG Operation failed roll back of a transaction in progress that encountered an error
56	Failed SERVINV update or could not get node name and number
57	Failed change to GWCEM registration
58	Failed to update NV
59	Failed RollBack of Change GWC
60	Associate MG Operation Preprovisining Failed
61	Associate MG Operation Preprovisioning is in progress
62	Failed MG9000 VMG line conversion
63	Upgrade MG Operation progress message indication
64	Change MG Operation Preprovisioning Failed (i.e. failed to add/delete line card(s))
65	The GWC is not deletable from GWCEM (i.e. provisioned data associated with it)
66	The MG is not deletable from GWCEM (i.e provisioned data associated with it)
67	The MG cannot be associated because the number of gateways in the call server is at the limit that a redirecting GWC can support.
68	Associate MG operation failed and was rolled back
69	The MG and GWC TN's are provisioned, LEN provisioning is incomplete retry to continue and complete LEN provisioning
70	The query of XaCore table BEARNETS failed
71	The query of XaCore table BEARNETS returned an empty list
72	An event was received by a state which it was able to process. This is a software error

6.7 Nodes Provisioning support for Internet Transparency

Provisioning Media Gateways in IP address spaces that are different to those in the Telecom Service Providers office is supported by the Internet Transparency capabilities of the OSSGate Nodes provisioning interface. This provides the ability to associate Media Gateways with NAT middleboxes as part of the Media Gateway provisioning requests. It also provides the ability to query the NAT middleboxes and Media Proxies that have been provisioned for a CS2000.

Provisioning Media Gateways that are behind a limited bandwidth link (LBL) is supported by the VCAC capabilities of the Nodes provisioning interface. This provides the ability to associate Media Gateways with LBL middleboxes as part of the Media Gateway provisioning requests.

Media Gateways which support service type 'ITRANS_ROAM' may be associated with up to five 'Root' middleboxes as part of the Media Gateway provisioning request. This allows terminals belonging to the Media Gateway to operate from behind NATs and limited bandwidth links.

Beginning with the SN08 release and introduction of support for Network Virtual Call Admission Control (VCAC) counting using the Session Policy Controller (SPC) platform, new network zone interfaces are introduced for use in situations where a Session Policy Controller is part of the deployment. These new interfaces are replacements for the current NAT and LBL provisioning interfaces, although the later remain in place and supported. Note that any topology (Network Zone) provisioning commands (i.e. add, change, delete) **MUST** be carried out in **both** the CS2000 Management system **AND** the Session Policy Controller. Each command should be sent first to the CS2000 Management system and, if this succeeds, to the Session Policy Controller (which supports the same XML schema).

The interface also provides the ability to provision the following network devices, and topological links between middleboxes.

- NAT and LBL middleboxes
- NAT, LBL and Composite NAT plus LBL network zones.
- Resource Usage profile (used by VCAC LBL middleboxes)
- Media Proxies

In order to support the functionality which allows SIP-T trunks to be configured as 'intra-domain' SIP-T trunks, provisioning of a unique identifier for each CS2K, the 'Call Agent Identifier' is provided. It

should be noted that the SN08 version of the SPC has no intra-domain support and hence this functionality should not be used where external VCAC counting is enabled for use with the SPC.

The OSSGate interface provides the following:

- Querying the Call Agent Identifier
- Setting the Call Agent Identifier
- Adding a NAT or LBL Middle box (modified for SN08)
- Changing a NAT or LBL middlebox attribute (modified for SN08)
- Deleting a NAT or LBL Middle box
- Adding a RU Profile
- Changing an RU profile attribute
- Deleting an RU profile
- Association of a MG to an adjacent NAT / LBL middlebox OR a set of root Middleboxes as part of the MG to Gateway Controller association request (see section 6.1.4 for definitions, examples provided below)
- Changing of the adjacent NAT / LBL middlebox OR set of rootMiddleboxes associated with a Media Gateway (modified for SN08)
- Listing of the adjacent NAT / LBL middlebox OR rootMiddleboxes assigned to a MG as part of the query of the MG data (see section 6.1.7 for definitions examples provided below)
- The ability to specify that the MG is outside Telecom Service Provider/TSP domain, but not behind a NAT or LBL, as part of the MG to Gateway Controller association request.
- Listing of all NAT or LBL middleboxes
- Listing of all NAT or LBL middleboxes associated with a specified Gateway Controller
- Listing of parameters for a specified NAT or LBL middlebox
- Listing of all Media Proxies
- Listing of all Media Proxies associated with a specified Gateway Controller
- Listing of parameters for a specified Media Proxy
- Add a NAT, LBL or Composite NAT plus LBL network zone (new for SN08).
- Delete a NAT, LBL or Composite NAT plus LBL network zone (new for SN08).

- Change the parent network zone associated with a NAT, LBL or Composite NAT plus LBL network zone (new for SN08).
- Query an individual network zone (new for SN08).
- Query all network zones (new for SN08).
- Query a range of network zones (new for SN08).

Note: All the above functions are also supported by the CS2000 Management Tools Graphical User Interface.

In addition to the above described interfaces, the following SPC interfaces utilized for external VCAC counting are known to OSSGate even though the functionality they describe is unsupported. The interfaces are known to OSSGate due to a desire to support a common XSD between the SPC and the CS2000 management system. Any attempts to use these interface will result in error 306 being returned to indicate this.

- Add a link layer type.
- Delete a link layer type.
- Change a link layer type
- Query a link layer type.

6.7.1 Terminology and Description of Method Parameters

- **version** - The version refers to the version of the method or operation - e.g., the addNAT operation may have version 1.0 and 2.0, where version 1.0 takes parameters X and Y and version 2.0 takes parms X, Y, and Z. TYPE *decimal*.
- **usn** - Unique Sequence Number. This parameter is introduced to facilitate sending multiple XML commands in one file. This parameter value will be used to match the responses with the requests. TYPE *integer*.
- **CallAgentId** - The unique integer that has been assigned by the user to this Call Agent (e.g. this CS2k). This must be a value between 1 and 255. TYPE *integer*.
- **CounterGWC** - The VCAC counter GWC IP address. TYPE *IPAddressType*
- **GWCname** - The name of the GWC. String of form GWC- followed by an integer 0-255 range.

- **IPAddress** - The Media Proxy IP address.
- **ListNATid** - Indication of whether the NAT id should be included in the response to this command. Set to 'true' if the NATid is required and 'false' if not.
- **itransMiddleboxName** - The name of the adjacent Middlebox (either NAT or LBL) for the gateway command. 32 character token without spaces.
- **LBLname** - The name of the LBL that this command applies to. 64 character token without spaces. *Type NoNameSpaceType.*
- **LBLid** - The unique middlebox id for this LBL. This must be an integer between 2 and 16777215. NB the allowed range for LBLid is smaller than the allowed range for the data type *MiddleBoxIndexType*.
- **MaxCount** - The VCAC count maximum limit. *TYPE RUMaxCountType.*
- **MPname** - The name of the Media Proxy that this command applies to. *TYPE NoSpaceNameType* (32 character token without spaces, see xsd definitions in Schema document, section 2.14)
- **NATname** - The name of the NAT that this command applies to. *TYPE NoSpaceNameType.*
- **NATid** - The unique middlebox id for this NAT. This must be an integer between 2 and 16777215. *TYPE MiddleBoxIndexType.* The allowed range for NATid is smaller than the allowed range for the data type *MiddleBoxIndexType*.
- **NATType** - Indicates the type of NAT, in particular, NATType = 1 indicates an uncontrolled NAT. No other values are used in this release. *TYPE string.*
- **ParentMB** - The name of the middlebox that is the parent of the NAT. 64 character token without spaces. *Type NoSpaceNameType.*
- **Protocol** - The Media Proxy communication protocol (GWC to MP). This must be either MGCP+ or MPCP. *TYPE ProtocolType.*
- **ProtocolVersion** - The Media Proxy communication protocol version. Version is restricted to a value of 2.0 for MGCP+ or a value of 3.0 for MPCP. *TYPE ProtocolVersionType.*

- **rootMiddleboxName1(1..5)** - The Middlebox name(s) of the root node middlebox(es) that encompasses the enterprise network that the CICM gateway operates with 64 character token without spaces
- **ReturnCode value**- A numeric value indicating the result of the attempted operation. TYPE *string*.
- **Return code text**- Appropriate text that describes the result of the operation. TYPE *string*.
- **RUDescription** - a text description for the RU Profile (maximum length of 64 characters). TYPE *string*.
- **RUValue** - a composite of the following parameters. Codec,PacketRate,Value. TYPE *RUValueParamsType*
- **Service** - Describes the service type supported by a network zone that is not an LBL. Type *NZServiceType*.
- **ID** - Indicates the ID for a given network zone. Type *NZIDType* (*unsigned integer minimum value = 2*). If it is not specified it will be generated.
- **Name** - Describes the name for a given network zone. Type *DNSNameType* (*string conforming to DNS naming conventions, maximum length = 32*).
- **Parent** - describes a network zone parent. Defined as a choice allowing the parent to be described by **ParentID** or **ParentName**. Type *ParentType*.
- **ParentID** - choice member of **ParentType**. Describes a parent network zone based on its ID. Type *unsignedInt*.
- **ParentName** - choice member of **ParentType**. Describes a parent network zone based on its name. Type *DNSNameType*.
- **IntraZoneBWInfo** - Composite of **BWCIR** and **NNSC**. These fields are only relevant to the SPC and will be ignored by OSSGate but must be correctly populated in accordance with the XSD interface if present. Type *BWInfoType* (*addNetworkZone*) **or** *IntroZoneBWInfoType* (*changeNetworkZone*).
- **LogicalNetworkLink** - Composite of **BandwidthInfo**, **LayerInfo** and **delete** (*changeNetworkZone* only). These fields are only relevant to the SPC and will be ignored by OSSGate, other than to indicate that the network zone possess LBL characteristics. When present it must be correctly populated. Type *AddLogicalLinkType*

(*addNetworkZone*) **or** *ChangeLogicalLinkType*
(*changeNetworkZone*).

- **IDMin** - filter describing the minimum network zone ID to be returned as part of a query operation. *Type unsignedInt.*
- **IDMax** - filter describing the maximum network zone ID to be returned as part of a query operation. *Type unsignedInt.*
- **MaxZones** - filter describing the maximum number of network zones to be returned as part of a query operation. *Type unsignedInt.*
- **NetworkZone** - Composite of **ID**, **Name** and **ParentID**.
- **Service** - Identifies Network Zone as being a NAT or Composite type Network Zone. For a Composite Network Zone the LogicalNetworkLink tag should also be present.
- **algName** - The name of the ALG node middlebox. *TYPE NoSpaceNameType* (32 character token without spaces)
- **ALGipaddress** The ALG IP address.
- **ALGPort** The Port used by ALG. Integer data type 0-65535
- **ALGProtocol**. The protocol used. Only valid value is NCS.

6.7.2 Description of Method Parameters

Method parameters are defined below. Input data is mandatory except where indicated otherwise.

1. **queryCallAgentId** - Method that gets the current value of the Call Agent Id for this Call Agent (CS2k). This method has the following parameters:
 - Input data:
 - usn
 - version
 - Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)

- CallAgentId - The value of the Call Agent Id for this Call Agent (CS2k).
 - ReturnCode - indicates via an integer value if the command has been successful or, if not, the error type and includes a brief textual message with further information.
- 2. **setCallAgentId** - Method that sets to Call Agent Id for this Call Agent (CS2k) to the specified value. This method has the following parameters:
 - Input data:
 - usn
 - version
 - CallAgentId - The desired value for the Call Agent Id for this Call Agent (CS2k).
 - Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - ReturnCode - indicates via an integer value if the command has been successful or, if not, the error type and includes a brief textual message with further information.
- 3. **queryNAT** - Method that gets the information regarding a NAT or set of NATs, depending on the parameters specified. If neither a NATname nor a GWCname is included, then information is returned for ALL NATs provisioned in this CS2k Mgr. This method has the following parameters:
 - Input data:
 - usn
 - version
 - NATname (optional, cannot be included as well as GWCname) - indicates the NAT for which information should be returned
 - GWCname (optional, cannot be included as well as NATname) - returns information for all NATs associated with this GWC
 - ListNATid (optional) - if set to true, this returns the NAT id as part information for each NAT, if

set to false or if the parameter is omitted the NAT id is not returned.

- Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - NATname
 - NATType (returned only when the NATname is specified)
 - NATid (returned only when ListNATid is set to 'true' in the input data)
 - ParentMB (if the NAT has no parent middlebox, the parent middlebox tag is empty. e.g. <ParentMB></ParentMB>). returned only when the NATname is specified OR ListNATid is set to 'true'

4. addNAT - Method that provision a NAT into the CS2k Mgr. This method has the following parameters: If the NAT id is not specified then the CS2k Mgr generates a NAT middlebox id for the NAT.

- Input data:
 - usn
 - version
 - NATname - unique name for the NAT to be provisioned
 - NATid (optional) - indicates the middlebox id that should be allocated to this NAT.
 - ParentMB (optional) -indicates that the NAT has a parent middlebox. The parent middlebox specified here must already have been provisioned into the CS2k Mgr.
- Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - ReturnCode

5. deleteNAT - Method that removes a NAT from the CS2k Mgr. This method has the following parameters:

- Input data:
 - usn
 - version
 - NATname - unique name for the NAT to be removed
 - Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - ReturnCode
6. **changeNAT** - Method that modifies the provisioned details of a NAT in the CS2k Mgr. This method has the following parameters:
- Input data:
 - usn
 - version
 - NATname - unique name identifying the NAT to be changed
 - ParentMB (optional) -indicates that the NAT has a parent middlebox. The parent middlebox specified here must already have been provisioned into the CS2k Mgr.
 - Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - ReturnCode
7. **queryLBL** - Method that gets the information regarding a LBL or set of LBLs, depending on the parameters specified. If neither a LBLname nor a GWCname is included, then information is returned for ALL LBLs provisioned in this CS2k Mgr. This method has the following parameters:
- Input data:
 - usn
 - version

- LBLname (optional, cannot be included as well as GWCname) - indicates the LBL for which information should be returned
 - GWCname (optional, cannot be included as well as LBLname) - returns information for all LBLs associated with this GWC
- Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - LBLname
 - LBLType (returned only when the LBL name is specified)
 - The Counter GWC IPAddress
 - The RU Description
 - Max Count Value
 - ParentMB (an empty value is returned if there is no parent Middlebox.
e.g. <ParentMB></ParentMB>).

8. addLBL - Method that provision a LBL into the CS2k Mgr. This method has the following parameters:

- Input data:
 - usn
 - version
 - LBLname - unique name for the LBL to be provisioned
 - LBLid (optional) - indicates the middlebox id that should be allocated to this LBL.
 - The Counter GWC IPAddress (optional).
 - The RU Description - must match a description of an RU Profile already provisioned into the CS2k Mgr.
 - Max Count Value.
 - ParentMB (optional) -indicates that the LBL has a parent middlebox. The parent middlebox specified here must already have been provisioned into the CS2k Mgr.

- Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - ReturnCode
- 9. **deleteLBL** - Method that removes a LBL from the CS2k Mgr. This method has the following parameters:
 - Input data:
 - usn
 - version
 - LBLname - unique name for the LBL to be removed
 - Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - ReturnCode
- 10. **changeLBL** - Method that modifies the provisioned details of a LBL in the CS2k Mgr. This method has the following parameters:
 - Input data:
 - usn
 - version
 - LBLname - unique name identifying the LBL to be changed.
 - The Counter GWC IPAddress (optional).
 - The RU Description (optional) - must match a description of an RU Profile already provisioned into the CS2k Mgr.
 - Max Count Value (optional).
 - ParentMB (optional) -indicates that the NAT has a parent middlebox. The parent middlebox specified here must already have been provisioned into the CS2k Mgr.
 - Output data:
 - usn (value should be the same as the input)

- version (value should be the same as the input)
- ReturnCode

11. queryMP - Method that gets the information regarding a MP or set of MPs, depending on the parameters specified. If neither a MPname nor a GWCname is included, then information is returned for ALL MPs provisioned in this CS2k Mgr. This method has the following parameters:

- Input data:
 - usn
 - version
 - MPname (optional, cannot be included as well as GWCname) - indicates the MP for which information should be returned
 - GWCname (optional, cannot be included as well as MPname) - returns information for all MPs associated with this GWC
- Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - MPname
 - IPAddress
 - Protocol
 - Protocol version

12. addMP - Method that provision a MP into the CS2k Mgr. This method has the following parameters:

- Input data:
 - usn
 - version
 - MPname - unique name for the NAT to be provisioned
 - IPAddress - the address must not be assigned already to another MP in the CS2k Mgr
 - Protocol
 - Protocol version

- Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - ReturnCode

13. deleteMP - Method that removes a MP from the CS2k Mgr. This method has the following parameters:

- Input data:
 - usn
 - version
 - MPname - unique name for the MP to be removed
- Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - ReturnCode

14. changeMP - Method that modifies the provisioned details of a NAT in the CS2k Mgr. This method has the following parameters:

- Input data:
 - usn
 - version
 - MPname - unique name identifying the MP to be changed
 - IPAddress - the new address must not be assigned already to another MP in the CS2k Mgr
 - Protocol
 - Protocol version
- Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - ReturnCode

15. changeRootMiddleboxes - Method that changes the rootMiddleboxes for a given gateway. This method has the following parameters:

- Input data:
 - usn
 - version
 - MGname indicates the gateway that this change command is for.
 - rootMiddleboxName1 - defines the 1st Root Middlebox.
 - rootMiddleboxName2 - (Optional) defines the 2nd Root Middlebox.
 - rootMiddleboxName3 - (Optional) defines the 3rd Root Middlebox.
 - rootMiddleboxName4 - (Optional) defines the 4th Root Middlebox.
 - rootMiddleboxName5 - (Optional) defines the 5th Root Middlebox.
- Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - ReturnCode - indicates via an integer value if the command has been successful or, if not, the error type and includes a brief textual message with further information.

16. addNetworkZone - Method that adds a new network zone of type NAT, LBL or Composite (containing the attributes of both a NAT and LBL simultaneously).

- Input data:
 - usn
 - version
 - ID - (Optional) defines the ID for a new network zone. The Call Agent ID must be defined for this parameter to be passed as valid.
 - Name - defines the name of a new network zone.
 - Service - (Optional) defines the service type supported by this network zone. Only NAT is

supported. The presence of Service defines the network zone as being either a NAT or Composite type.

- Parent - (Optional) defines the parent network zone associated with the new network zone. Parent network zone may be described by either the ParentName (network zone Name) or ParentID (network zone ID) tags.
- IntraZoneBWInfo - (Optional) the attributes contained within IntraZoneBWInfo are only of relevance to the SPC and will be ignored by OSSGate if supplied.
- LogicalNetworkLink - (Optional) the presence of this tag is interpreted as indicating that the network zone is of type LBL or Composite (in conjunction with Service). The actual parameters within the LogicalNetworkLink are only of relevance to the SPC and will be ignored by OSSGate if supplied.

- Output data:

- usn (value should be the same as the input)
- version (value should be the same as the input)
- ReturnCode - indicates via an integer value if the command has been successful or, if not, the error type and includes a brief textual message with further information.

17. **changeNetworkZone** - Method that modifies and existing network zone.

Note 1: Only the associated parent network zone may be modified by this interface. This is a subset of the functionality supported by the SPC and the XSD itself. Operations that are unsupported by OSSGate but legal for the SPC will be rejected.

- Input data:

- usn
- version
- ID - (Optional) describes the ID of the network zone to be changed. Request rejected if specified in conjunction with Name as modification of a network zone name is currently unsupported by OSSGate.

- Name - (Optional) describes the name of the network zone to be modified.
 - Parent - (Optional) defines the parent network zone associated with the new network zone. Parent network zone may be described by either the ParentName (network zone Name) or ParentID (network zone ID) tags.
 - IntraZoneBWInfo - (Optional) the attributes contained within IntraZoneBWInfo are only of relevance to the SPC and will be ignored by OSSGate if supplied.
 - LogicalNetworkLink - (Optional) the attributes contained within the LogicalNetworkLink are only of relevance to the SPC and will have no impact on OSSGate. However checking is imposed to detect a change of type to/from NAT and Composite for this network zone as this is unsupported functionality for OSSGate. Attempts to change type will be rejected to prevent OSSGate / SPC mismatch.
- Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - ReturnCode - indicates via an integer value if the command has been successful or, if not, the error type and includes a brief textual message with further information.

18. deleteNetworkZone - Method that deletes an existing network zone.

- Input data:
 - usn
 - version
 - ID - (Optional) describes the ID of the network zone to be deleted (ID or Name must be specified).
 - Name - (Optional) describes the name of the network zone to be deleted (ID or Name must be specified).
- Output data:
 - usn (value should be the same as the input)

- version (value should be the same as the input)
- ReturnCode - indicates via an integer value if the command has been successful or, if not, the error type and includes a brief textual message with further information.

19. queryNetworkZone - Method that query one or more network zones.

- Input data:

- usn
- version
- ID - (Optional) describes the ID of the network zone to be queried. Mutually exclusive with Name. Restricts query to a single unique network zone.
- Name - (Optional) describes the name of the network zone to be queried. Mutually exclusive with ID. Restricts query to a single unique network zone.
- IDMin - (Optional) acts as a filter to restrict return data from a query. Specifies the minimum ID for a network zone to be returned.
- IDMax - (Optional) acts as a filter to restrict the return data from a query. Specifies the maximum ID for a network zone to be returned.
- MaxZones - (Optional) acts as a filter to restrict the return data from a query. Specifies an upper limit on the number of network zones that may be returned.

- Output data:

- usn (value should be the same as the input)
- version (value should be the same as the input)
- ReturnCode - indicates via an integer value if the command has been successful or, if not, the error type and includes a brief textual message with further information.
- ID - the ID for the returned network zone. Contained within a NetworkZone tag where query targets multiple network zones (ID / Name tags not present in input data).

- Name - the name of the returned network zone. Contained within a NetworkZone tag where query targets multiple network zones.
- ParentID - the ID of the parent network zone (0 if no parent). Contained within a NetworkZone tag where query targets multiple network zones.
- Service - (Optional) returned where the network zone is of type NAT / Composite *and* the query targets an individual network zone by specifying the Name / ID tags.
- LogicalNetworkLink - (Optional) returned where the network zone is of type LBL / Composite *and* the query targets an individual network zone. The LogicalNetworkLink is data filled with dummy data as OSSGate has no knowledge of SPC specific data.

6.7.3 SPC Supported Interfaces

The SPC platform supports a subset of the Internet Transparency interfaces that are supported via OSSGate for the CS2000 management system. Those interfaces only supported by the CS2000 management system should not be sent to the SPC. The following interfaces are supported by both the SPC and OSSGate:

- addNetworkZone
- changeNetworkZone
- deleteNetworkZone
- queryNetworkZone

In addition the following interfaces are supported by the SPC but not from OSSGate for the CS2000 management system:

- addLinkLayerType
- changeLinkLayerType
- deleteLinkLayerType
- queryLinkLayerType

6.7.4 Query Call Agent Identifier Command

Figure 40 XML command queryCallAgentId

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryCallAgentId usn="1" version="1.0"/>
    </Methods>
  </Command>
</CommandList>
```

Figure 41 XML Response for queryCallAgentId

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryCallAgentId usn="1" version="1.0">
        <ReturnData>
          <CallAgentId>3</CallAgentId>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </queryCallAgentId>
    </Methods>
  </Response>
</CommandList>
```

6.7.5 Set Call Agent Identifier command

Figure 42 XML Command: setCallAgentId

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <setCallAgentId usn="1" version="1.0">
        <Parameters>
          <CallAgentId>3</CallAgentId>
        </Parameters>
      </setCallAgentId>
    </Methods>
  </Command>
</CommandList>
```

Figure 43 XML Response for setCallAgentId

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <setCallAgentId usn="1" version="1.0">
        <ReturnData>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </setCallAgentId>
    </Methods>
  </Response>
</CommandList>
```

6.7.6 Associate Gateway with Adjacent Middlebox Command

The association MG to adjacent MB is only allowed if the gateway supports the Internet Transparency service type (ITRANS). The OSS can associate a Media Gateway with a Middle Box and specify if the gateway is inside or outside the Telecoms Service Providers (IN/OUT TSP) domain. This is used to determine if the Media Proxy NAT services are required or the VCAC counting service should be enabled for a call flow. A specific Middlebox "outtsp" is reserved to indicate that the MG is outside the TSP/Succession Site but NOT behind a NAT. An additional field is added to the existing *assocMG* method XML query. The new element `<itransMiddleboxName>` is optional. When used, it must contain the name of an existing NAT or LBL middle box.

Figure 44 An example of assocMG command with adjacent Middlebox addition

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
<Command>
  <Interface>cs2kCfgMgrIf</Interface>
  <Methods>
    <assocMG usn="1" version="1.0">
      <Parameters>
        <mgUIName>mg4.nortel.net</mgUIName>
        <mgProfileName>ASKEY_LINE_GW_4</mgProfileName>
        <mgIpAddr>47.49.23.48</mgIpAddr>
        <mgProtocolType>5</mgProtocolType>
        <mgProtocolVersion>2.0</mgProtocolVersion>
        <mgProtocolPort>2427</mgProtocolPort>
        <gwcUIName>GWC-0</gwcUIName>
        <itransMiddleboxName>myNat1</itransMiddleboxName>
      </Parameters>
    </assocMG>
  </Methods>
</Command>
```

Figure 45 An example of assocMG Response with adjacent Middlebox addition:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
<Response>
  <Interface>cs2kCfgMgrIf</Interface>
  <Methods><assocMG usn="1" version="1.0">
    <ReturnData>
      <Row>
        <gwcUIName>GWC-0</gwcUIName>
        <gwclpAddr>172.17.40.24</gwclpAddr>
        <mgNodeName>LG 00 0</mgNodeName>
      </Row>
      <RC>0</RC>
      <MsgTxt>The MG was successfully associated with a GWC</MsgTxt>
    </ReturnData>
  </assocMG>
</Methods>
</Response>
</CommandList>
```

If the Media Gateway to be added is outside the succession VPN and not behind a NAT or LBL, the itransMiddleboxName XML tag must be used with a value of “*outtsp*”.

6.7.7 QueryMG Command

An additional field is added to the existing *QueryMG* method XML response. The new element <itransMiddleboxName> is returned when a NAT or LBL

middle box is associated with the gateway. When no NAT or LBL is associated to the gateway, an empty tag is returned.

Figure 46 An example of queryMG command response with NAT

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Response>
    <Interface>cs2kCfgMgrIf</Interface>
    <Methods>
      <queryMG usn="1" version="1.0">
        <ReturnData><Row><mgUIList>mg4.nortel.net</mgUIList>
          <mgIpList>47.49.23.48</mgIpList>
          <callServerId>RTPS</callServerId>
          <gwcUIName>GWC-0</gwcUIName>
          <svcTypeList>0</svcTypeList>
          <nodeName>LG 00 0</nodeName>
          <itransMiddleboxName>myNat1</itransMiddleboxName>
          <engrEndPoints>4</engrEndPoints>
          <protType>mgcp</protType>
          <protVersion>2.0</protVersion>
          <protPort>2427</protPort>
          <profileName>ASKEY_LINE_GW_4</profileName>
        </Row><RC>0</RC>
        <MsgTxt>Query of a Single MG was successful</MsgTxt>
      </ReturnData>
    </queryMG>
  </Methods>
</Response>
</CommandList>
```

Note: When the MG is outside the succession VPN and not behind a NAT, the itransMiddleboxName value is set to “*outtsp*” (‘outside telecom service provider VPN’).

6.7.8 Associate Gateway with Root Middleboxes Command

The association MG to root MBs is only allowed if the gateway supports the Internet Transparency service type (ITRANS_ROAM). This is used to determine if the Media Proxy NAT services are required or the VCAC counting service should be enabled for a call flow. Additional fields are added to the existing *assocMG* method XML query, these are defined in section 6.1.4. The new elements <rootMiddleboxName1>, <rootMiddleboxName2>, <rootMiddleboxName3>, <rootMiddleboxName4> & <rootMiddleboxName4> are optional. When used, they must contain the name of an existing NAT or LBL middle box.

Figure 47 An example of assocMG command with root Middleboxes addition

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>cs2kCfgMgrIf</Interface>
    <Methods>
      <assocMG usn="1" version="1.0">
        <Parameters>
          <mgUIName>CICM-022</mgUIName>
          <mgProfileName>CICM</mgProfileName>
          <mgIpAddr>11.13.45.58</mgIpAddr>
          <mgProtocolType>4</mgProtocolType>
          <mgProtocolVersion>1.0</mgProtocolVersion>
          <mgProtocolPort>2944</mgProtocolPort>
          <gwcUIName>GWC-107</gwcUIName>
          <reservedTerminations>1023</reservedTerminations>
          <rootMiddleboxName1>nat1</rootMiddleboxName1>
          <rootMiddleboxName2>nat2</rootMiddleboxName2>
          <rootMiddleboxName3>nat3</rootMiddleboxName3>
          <rootMiddleboxName4>nat4</rootMiddleboxName4>
          <rootMiddleboxName5>lbl1</rootMiddleboxName5>
        </Parameters>
      </assocMG>
    </Methods>
  </Command>
</CommandList>
```

6.7.9 QueryMG Command (Root Middleboxes Returned)

Additional fields are added to the existing *QueryMG* method XML response. The new root Middlebox Names are returned when root middleboxes are associated with the gateway. When no root Middleboxes are associated to the gateway, empty tags are returned.

Figure 48 An example of queryMG command response with root MBs

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList><Response><Interface>cs2kCfgMgrIf</Interface>
<Methods><queryMG usn="1" version="1.0">
<ReturnData><Row>
<mgUIList>CICM-322</mgUIList>
<mgIpList>11.13.45.58</mgIpList>
<callServerId>telco</callServerId>
<gwcUIName>GWC-222</gwcUIName>
<svcTypeList>0</svcTypeList><svcTypeList>7</svcTypeList>
<nodeName>LG</nodeName>
<engrEndPoints>1023</engrEndPoints>
<protType>megaco</protType>
<protVersion>1.0</protVersion>
<protPort>2944</protPort>
<profileName>CICM</profileName>
<maxEngrEndPoints>3069</maxEngrEndPoints>
<rootMiddleboxName1>lbl1</rootMiddleboxName1>
<rootMiddleboxName2>nat1</rootMiddleboxName2>
<rootMiddleboxName3>nat2</rootMiddleboxName3>
<rootMiddleboxName4></rootMiddleboxName4>
<rootMiddleboxName5></rootMiddleboxName5>
</Row><RC>0</RC>
<MsgTxt>Query of a Single MG was successful</MsgTxt>
</ReturnData></queryMG>
</Methods></Response></CommandList>
```

6.7.10 Add NAT Command

A NAT can be added to the CS2000 management system by using a unique name.

Figure 49 An example of AddNAT command request

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
<Command>
<Interface>ITransIf</Interface>
<Methods>
<addNAT usn="1" version="1.0">
<Parameters>
<NATName>Nat-B</NATName>
<NATId>12345</NATId>
<ParentMB>Nat-A</ParentMB>
</Parameters>
</addNAT>
</Methods>
</Command>
</CommandList>
```

Figure 50 An example of AddNAT response

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <addNAT usn="1" version="1.0">
        <ReturnData>
          <ReturnCode value="0" text="Successful result"/>
        </ReturnData>
      </addNAT>
    </Methods>
  </Response>
</CommandList>
```

Note: Attempting to add a NAT name already added to the system will fail with a message "Failed to add the Middlebox because Name specified is already in use: Nat-B"

Depending on the parameters used in the XML query the response returns one of the following:

- Success result
- NAT name already in use.
- NAT id already in use
- Unknown Parent Middlebox

6.7.11 Delete NAT command

A provisioned NAT can be deleted from the CS2000 management system by passing the NAT name. A existing NAT will be deleted only if the NAT has no association to a media gateway.

Figure 51 An example of deleteNAT command request

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <deleteNAT usn="1" version="1.0">
        <Parameters>
          <NATname>NAT-A</NATname>
        </Parameters>
      </deleteNAT>
    </Methods>
  </Command>
</CommandList>
```

Figure 52 An example of deleteNAT response

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <deleteNAT usn="1" version="1.0">
        <ReturnData>
          <ReturnCode value="0" text="Successful result"/>
        </ReturnData>
      </deleteNAT>
    </Methods>
  </Response>
</CommandList>
```

Depending on the parameters used in the XML query the response returns:

- Success result
- MiddleBox name not found

Note 1: If a NAT is provisioned into multiple CS2ks, care should be taken when deleting the NAT. If the NAT middlebox ID used for the shared NAT has been automatically generated by one CS2k Mgr, the NAT should not be deleted from that CS2k Mgr while it is still provisioned in other CS2ks. It must be deleted from all other CS2k Mgrs first.

6.7.12 Change NAT command

A provisioned NAT can be changed on the CS2000 management system by passing the NAT name and a new parent middlebox name.

Figure 53 XML command to Change specified NAT parent middlebox

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <changeNAT usn="1" version="1.0">
        <Parameters>
          <NATname>NAT-A</NATname>
          <ParentMB>VPN-Eastenders</ParentMB>
        </Parameters>
      </changeNAT>
    </Methods>
  </Command>
</CommandList>
```

Figure 54 XML response to Change specified NAT

```

<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <changeNAT usn="1" version="1.0">
        <ReturnData>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </changeNAT>
    </Methods>
  </Response>
</CommandList>

```

Depending on the parameters used in the XML query the response returns:

- Success result
- MiddleBox name not found

6.7.13 Query NAT command

The *queryNAT* command returns:

- information concerning a given NAT
- names of the NAT used on a given GWC
- names of all NATs provisioned

The information regarding each NAT returned is as follows:

- NAT name
- NAT type - only in *queryNAT* request in which a single NAT is specified
- NAT parent middlebox. If the NAT has no parent middlebox, the parent middlebox tag is empty.
e.g. <ParentMB></ParentMB>).
- only in *queryNAT* request in which a single NAT is specified OR if ListNATid tag is "true"

An additional optional tag may be specified in the queryNAT command, the 'ListNATid' tag. If this tag is present and has a value true, the NAT middlebox identifier is also listed for each NAT returned. This tag may be present in all the forms of the queryNAT command i.e. for a specified NAT, on a given GWC and for all provisioned NATs). If the tag is not present, middlebox identifiers are not listed.

Figure 55 An example of query one NAT request:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryNAT usn="1" version="1.0">
        <Parameters>
          <NATname>myNat1</NATname>
        </Parameters>
      </queryNAT>
    </Methods>
  </Command>
</CommandList>
```

Figure 56 An example of query one NAT response

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryNAT usn="1" version="1.0">
        <ReturnData>
          <NATname>myNat1</NATname>
          <NATType>1</NATType>
          <ParentMB>parentNat</ParentMB>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </queryNAT>
    </Methods>
  </Response>
</CommandList>
```

Figure 57 An example of query one NAT request, with NAT id:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryNAT usn="1" version="1.0">
        <Parameters>
          <NATname>myNat1</NATname>
          <ListNATid>true</ListNATid>
        </Parameters>
      </queryNAT>
    </Methods>
  </Command>
</CommandList>
```

Figure 58 An example of query one NAT response, including NAT id:

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryNAT usn="1" version="1.0">
        <ReturnData>
          <NATName>myNat1</NATName>
          <NATType>1</NATType>
          <ParentMB>parentNat</ParentMB>
          <NATid>65538</NATid>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </queryNAT>
    </Methods>
  </Response>
</CommandList>
```

Figure 59 An example of query all NATs on a given GWC request, including NAT id

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryNAT usn="1" version="1.0">
        <Parameters>
          <GWCname>GWC-0</GWCname>
          <ListNATid>true</ListNATid>
        </Parameters>
      </queryNAT>
    </Methods>
  </Command>
</CommandList>
```


Figure 60 An example of query all NATs on a given GWC response including NAT id.

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryNAT usn="1" version="1.0">
        <ReturnData>
          <NATname>myNat1</NATname>
          <NATid>234</NATid>
          <ParentMB></ParentMB>
          <NATname>myNat2</NATname>
          <NATid>234</NATid>
          <ParentMB>aParentmb</ParentMB>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </queryNAT>
    </Methods>
  </Response>
</CommandList>
```

Figure 61 An example of query all NATs request:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryNAT usn="1" version="1.0">
        <Parameters>
          </Parameters>
        </queryNAT>
      </Methods>
    </Command>
  </CommandList>
```

Figure 62 An example of query all NATs response:

```

<?xml version='1.0'?>
<CommandList>
<Response>
<Interface>ITransIf</Interface>
<Methods>
<queryNAT usn="1" version="1.0">
<ReturnData>
<NATname>NAT1</NATname>
<NATname>NAT2</NATname>
<NATname>NAT3</NATname>
<NATname>NAT4</NATname>
<NATname>NAT5</NATname>
<ReturnCode value="0" text="Successful result" />
</ReturnData>
</queryNAT>
</Methods>
</Response>
</CommandList>

```

6.7.14 Add LBL command

A LBL can be added to the CS2000 management system by using a unique name.

Figure 63 XML command to Create a LBL device

```

<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
<Command>
<Interface>ITransIf</Interface>
<Methods>
<addLBL usn="1" version="1.0">
<Parameters>
<LBLname>LBL1</LBLname>
<CounterGWC>47.132.132.12</CounterGWC>
<RUDescription>AAL1/ip40</RUDescription>
<MaxCount>1000</MaxCount>
<ParentMB>VPN-A</ParentMB>
</Parameters>
</addLBL>
</Methods>
</Command>
</CommandList>

```

Figure 64 XML response to Creating a LBL device.

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <addLBL usn="1" version="1.0">
        <ReturnData>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </addLBL>
    </Methods>
  </Response>
</CommandList>
```

Depending on the parameters used in the XML query the response returns on of the following:

- Success result
- LBL already in use.
- LBL id already in use
- Unknown Parent Middlebox

6.7.15 Delete LBL command

A provisioned LBL can be deleted from the CS2000 management system by passing the LBL name. The existing LBL will be deleted only if the LBL has no association to a media gateway.

Figure 65 XML command to Delete a LBL device

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <deleteLBL usn="1" version="1.0">
        <Parameters>
          <LBLname>LBL1</LBLname>
        </Parameters>
      </deleteLBL>
    </Methods>
  </Command>
</CommandList>
```

Figure 66 XML response to Deleting a LBL device.

```

<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <deleteLBL usn="1" version="1.0">
        <ReturnData>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </deleteLBL>
    </Methods>
  </Response>
</CommandList>

```

Depending on the parameters used in the XML query the response returns:

- Success result
- MiddleBox name not found

6.7.16 Change LBL command

A provisioned LBL can be changed on the CS2000 management system by passing the LBL name.

Figure 67 XML command to Change a LBL device RU profile.

```

<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <changeLBL usn="1" version="1.0">
        <Parameters>
          <LBLname>LBL1</LBLname>
          <RUDescription>AAL2/ip40</RUDescription>
        </Parameters>
      </changeLBL>
    </Methods>
  </Command>
</CommandList>

```

Figure 68 XML response to Changing a LBL device.

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <changeLBL usn="1" version="1.0">
        <ReturnData>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </changeLBL>
    </Methods>
  </Response>
</CommandList>
```

Depending on the parameters used in the XML query the response returns:

- Success result
- MiddleBox name not found

6.7.17 Query LBL command

The *query LBL* command returns:

- LBL name
- The Counter GWC IPAddress
- The RU Description
- Max Count Value
- Parent MB name

Figure 69 XML command to Query a LBL

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryLBL usn="1" version="1.0">
        <Parameters>
          <LBLname>LBL-A</LBLname>
        </Parameters>
      </queryLBL>
    </Methods>
  </Command>
</CommandList>
```

Figure 70 XML response to Query LBL

```
<?xml version='1.0'?>
<CommandList>
<Response>
  <Interface>ITransIf</Interface>
  <Methods>
    <queryLBL usn="1" version="1.0">
      <ReturnData>
        <LBLname>LBL1</LBLname>
        <CounterGWC>47.132.132.12</CounterGWC>
        <RUDescription>AAL1/ip40</RUDescription>
        <MaxCount>1000</MaxCount>
        <ParentMB>VPN-A</ParentMB>
        <ReturnCode value="0" text="Successful result" />
      </ReturnData>
    </queryLBL>
  </Methods>
</Response>
</CommandList>
```

6.7.18 Add RU Profile command

A RU can be added to the CS2000 management system by using a unique RU Description, and RUValue which includes the Codec, PacketRate and Value.

Figure 71 XML command to Create an RU profile

```

<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <addRU usn="1" version="1.0">
        <Parameters>
          <RUDescription>AAL1/IP40</RUDescription>
          <RUValue>
            <Codec>G.711</Codec>
            <PacketRate>10 ms</PacketRate>
            <Value>103</Value>
          </RUValue>
          <RUValue>
            <Codec>G.711</Codec>
            <PacketRate>20 ms</PacketRate>
            <Value>133</Value>
          </RUValue>
          <RUValue>
            <Codec>G.729</Codec>
            <PacketRate>10 ms</PacketRate>
            <Value>132</Value>
          </RUValue>
          <RUValue>
            <Codec>G.729</Codec>
            <PacketRate>20 ms</PacketRate>
            <Value>103</Value>
          </RUValue>
        </Parameters>
      </addRU>
    </Methods>
  </Command>
</CommandList>

```

Figure 72 XML response to Creating an RU profile.

```

<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <addRU usn="1" version="1.0">
        <ReturnData>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </addRU>
    </Methods>
  </Response>
</CommandList>

```

Depending on the parameters used in the XML query the response returns on of the following:

- Success result
- RU already in use.

6.7.19 Delete RU Profile command

A provisioned RU can be deleted from the CS2000 management system by passing the RU name. A existing RU will be deleted only if the RU has no association to an LBL.

Figure 73 XML command to Delete an RU Profile

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <deleteRU usn="1" version="1.0">
        <Parameters>
          <RUDescription>AAL1/IP40</RUDescription>
        </Parameters>
      </deleteRU>
    </Methods>
  </Command>
</CommandList>
```

Figure 74 XML response to Deleting an RU profile.

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <deleteRU usn="1" version="1.0">
        <ReturnData>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </deleteRU>
    </Methods>
  </Response>
</CommandList>
```

Depending on the parameters used in the XML query the response returns:

- Success result
- RU name not found
- RU in use by an LBL

6.7.20 Change RU Profile command

A provisioned RU can be changed on the CS2000 management system by passing the RU name.

Figure 75 XML to Change RU profile values

```

<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <changeRU usn="1" version="1.0">
        <Parameters>
          <RUDescription>AAL1/IP40</RUDescription>
          <RUValue>
            <Codec>G.711</Codec>
            <PacketRate>10 ms</PacketRate>
            <Value>123</Value>
          </RUValue>
          <RUValue>
            <Codec>G.711</Codec>
            <PacketRate>20 ms</PacketRate>
            <Value>133</Value>
          </RUValue>
          <RUValue>
            <Codec>G.729</Codec>
            <PacketRate>10 ms</PacketRate>
            <Value>152</Value>
          </RUValue>
          <RUValue>
            <Codec>G.729</Codec>
            <PacketRate>20 ms</PacketRate>
            <Value>163</Value>
          </RUValue>
        </Parameters>
      </changeRU>
    </Methods>
  </Command>
</CommandList>

```

Figure 76 XML response to Changing an RU profile.

```

<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <changeRU usn="1" version="1.0">
        <ReturnData>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </changeRU>
    </Methods>
  </Response>
</CommandList>

```

Depending on the parameters used in the XML query the response returns:

- Success result
- RU name not found

6.7.21 Query RU Profile command

The *query RU* command returns:

- RU Description
- Codec
- PacketRate
- Value.

Figure 77 XML command to Query all RU Profiles

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryRU usn="1" version="1.0">
        <Parameters>
          </Parameters>
        </queryRU>
      </Methods>
    </Command>
  </CommandList>
```

Figure 78 XML response to Query all RU Profiles

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryRU usn="1" version="1.0">
        <ReturnData>
          <RUDescription>AAL1/IP40</RUDescription>
          <RUDescription>AAL2/IP40</RUDescription>
          <RUDescription>AAL1/IP40-B</RUDescription>
          <RUDescription>AAL2/IP40-B</RUDescription>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </queryRU>
    </Methods>
  </Response>
</CommandList>
```

Figure 79 XML command to Query RU Profile data

```

<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryRU usn="1" version="1.0">
        <Parameters>
          <RUDescription>AAL1/IP40</RUDescription>
        </Parameters>
      </queryRU>
    </Methods>
  </Command>
</CommandList>

```

Figure 80 XML response to Querying an RU profile.

```

<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryRU usn="1" version="1.0">
        <ReturnData>
          <RUDescription>AAL1/IP40</RUDescription>
          <RUValue>
            <Codec>G.711</Codec>
            <PacketRate>10 ms</PacketRate>
            <Value>123</Value>
          </RUValue>
          <RUValue>
            <Codec>G.711</Codec>
            <PacketRate>20 ms</PacketRate>
            <Value>133</Value>
          </RUValue>
          <RUValue>
            <Codec>G.729</Codec>
            <PacketRate>10 ms</PacketRate>
            <Value>152</Value>
          </RUValue>
          <RUValue>
            <Codec>G.729</Codec>
            <PacketRate>20 ms</PacketRate>
            <Value>163</Value>
          </RUValue>
        </ReturnData>
        <ReturnCode value="0" text="Successful result" />
      </queryRU>
    </Methods>
  </Response>
</CommandList>

```

6.7.22 Add MP command

A MP can be added to the CS2000 management system by using a unique name.

Figure 81 XML command to Create a new Media Proxy

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <addMP usn="1" version="1.0">
        <Parameters>
          <MPName>MP-A</MPName>
          <IPAddress>47.165.32.23</IPAddress>
          <Protocol>MGCP+</Protocol>
          <ProtocolVersion>2.0</ProtocolVersion>
        </Parameters>
      </addMP>
    </Methods>
  </Command>
</CommandList>
```

Figure 82 XML response to Creating a new Media Proxy command

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <addMP usn="1" version="1.0">
        <ReturnData>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </addMP>
    </Methods>
  </Response>
</CommandList>
```

Depending on the parameters used in the XML query the response returns on of the following:

- Success result
- MP already in use.

6.7.23 Delete MP command

A provisioned MP can be deleted from the CS2000 management system by passing the MP name. A existing MP will be deleted only if the MP has no association to a GWC gateway.

Figure 83 XML command to Delete a Media Proxy

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <deleteMP usn="1" version="1.0">
        <Parameters>
          <MPname>MP-A</MPname>
        </Parameters>
      </deleteMP>
    </Methods>
  </Command>
</CommandList>
```

Figure 84 XML response to Deleting Media Proxy.

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <deleteMP usn="1" version="1.0">
        <ReturnData>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </deleteMP>
    </Methods>
  </Response>
</CommandList>
```

Depending on the parameters used in the XML query the response returns:

- Success result
- MP name not found

6.7.24 Change MP command

A provisioned MP can be changed on the CS2000 management system by passing the MP name.

Figure 85 XML command to Change IP data of a Media Proxy

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <changeMP usn="1" version="1.0">
        <Parameters>
          <MPName>MP-A</MPName>
          <IPAddress>47.165.32.22</IPAddress>
          <Protocol>MGCP+</Protocol>
          <ProtVersion>2.0</ProtVersion>
        </Parameters>
      </changeMP>
    </Methods>
  </Command>
</CommandList>
```

Figure 86 XML response to Change Media Proxy IP data.

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <changeMP usn="1" version="1.0">
        <ReturnData>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </changeMP>
    </Methods>
  </Response>
</CommandList>
```

Depending on the parameters used in the XML query the response returns:

- Success result
- MP name not found

6.7.25 Query MP command

The *queryMP* command returns:

- complete information concerning a given MP
- names of the Media Proxies used on a given GWC
- names of all Media Proxies provisioned.

Figure 87 An example of query one Media Proxy request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryMP usn="1" version="1.0">
        <Parameters>
          <MPname>churchill</MPname>
        </Parameters>
      </queryMP>
    </Methods>
  </Command>
</CommandList>
```

Figure 88 An example of query one Media Proxy response

```
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryMP usn="1" version="1.0">
        <ReturnData>
          <MPname>churchill</MPname>
          <DeviceIP>1.1.1.1</DeviceIP>
          <Protocol>MGCP+</Protocol>
          <ProtocolVersion>2.0</ProtocolVersion>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </queryMP>
    </Methods>
  </Response>
</CommandList>
```

Figure 89 An example of query Media Proxies on a given GWC request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryMP usn="1" version="1.0">
        <Parameters>
          <GWName>GWC-102</GWName>
        </Parameters>
      </queryMP>
    </Methods>
  </Command>
</CommandList>
```

Figure 90 An example of query Media Proxies on a given GWC response:

```
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryMP usn="1" version="1.0">
        <ReturnData>
          <MPname>admiral</MPname>
          <MPname>churchill</MPname>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </queryMP>
    </Methods>
  </Response>
</CommandList>
```

Figure 91 An example of query all Media Proxies request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryMP usn="1" version="1.0">
        <Parameters>
        </Parameters>
      </queryMP>
    </Methods>
  </Command>
</CommandList>
```

Figure 92 An example of query all Media Proxies response:

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryMP usn="1" version="1.0">
        <ReturnData>
          <MPname>admiral</MPname>
          <MPname>churchill</MPname>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </queryMP>
    </Methods>
  </Response>
</CommandList>
```


6.7.26 Change Gateway's Associated NAT or LBL command

This method provides ability to change the adjacent NAT or LBL middle box associated with a gateway.

Figure 93 An example of Change middlebox associated to a GW request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <changeAssocMB usn="1" version="1.0">
        <Parameters>
          <MGname>NorthBrook1105.com</MGname>
          <itransMiddleboxName>myNAT1</itransMiddleboxName>
        </Parameters>
      </changeAssocMB>
    </Methods>
  </Command>
</CommandList>
```

Figure 94 An example of Change middlebox associated to a GW response:

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <changeAssocMB usn="1" version="1.0">
        <ReturnData>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </changeAssocMB>
    </Methods>
  </Response>
</CommandList>
```

To disassociate a gateway from a NAT or LBL, omit the itransMiddleboxName XML tag. This way, the gateway will not be associated to any NAT.

To change the status of a gateway as being outside the succession VPN and not behind a NAT or LBL, set the itransMiddleboxName XML tag to “*outtsp*”.

6.7.27 Change Gateway's Associated root Middleboxes command

This method provides ability to change the root middle boxes associated with a gateway.

Figure 95 An example of Change root Middleboxes associated to a GW request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <changeRootMiddleboxes usn="1" version="1.0">
        <Parameters>
          <MGname>CICM-001</MGname>
          <rootMiddleboxName1>nat1</rootMiddleboxName1>
          <rootMiddleboxName2>nat2</rootMiddleboxName2>
          <rootMiddleboxName3>nat3</rootMiddleboxName3>
        </Parameters>
      </changeRootMiddleboxes>
    </Methods>
  </Command>
</CommandList>
```

Figure 96 An example of Change root Middleboxes associated to a GW response:

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <changeRootMiddleboxes usn="1" version="1.0">
        <ReturnData>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </changeRootMiddleboxes>
    </Methods>
  </Response>
</CommandList>
```

To disassociate a gateway from root Middleboxes, omit the rootMiddleboxName1..5 XML tags. This way, the gateway will not be associated to any root Middleboxes.

6.7.28 Add Network Zone Command

A Network zone can be added to the CS2000 management system by using a unique name and optionally an ID.

Network Zones should added to the system in situations there is a Session Policy Controller deployed and commissioned and where Network VCAC has been enabled.

In order for a Network Zones ID to be specified at time of creation, the Call Agent ID should be setup within the CS200 management system. As the SPC platform requires the ID to be specified it is recommended that this be done prior to the creation of network zones.

Network zones may be of type NAT, LBL and Composite (both NAT and LBL characteristics). XML examples are given for all variations:

Figure 97 An example of Add Network Zone (NAT type) request:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <addNetworkZone usn="1" version="1.0">
        <Parameters>
          <ID>234</ID>
          <Name>router1</Name>
          <Service>NAT</Service>
          <Parent>
            <ParentID>123</ParentID>
          </Parent>
        </Parameters>
      </addNetworkZone>
    </Methods>
  </Command>
</CommandList>
```

Figure 98 An example of Add Network Zone (LBL type) request:

```

<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <addNetworkZone usn="1" version="1.0">
        <Parameters>
          <ID>235</ID>
          <Name>router2</Name>
          <Parent>
            <ParentName>router1</ParentName>
          </Parent>
          <IntraZoneBWInfo>
            <BWCIR>321</BWCIR>
            <NNSC>
              <Premium>12</Premium>
              <Platinum>23</Platinum>
              <Gold>25</Gold>
              <Silver>3</Silver>
              <Bronze>5</Bronze>
            </NNSC>
          </IntraZoneBWInfo>
          <LogicalNetworkLink>
            <BandWidthInfo>
              <Symmetry>false</Symmetry>
              <UpBWInfo>
                <BWCIR>456</BWCIR>
              </UpBWInfo>
              <DownBWInfo>
                <BWCIR>997</BWCIR>
              </DownBWInfo>
            </BandWidthInfo>
            <LayerInfo>
              <Layer>
                <LinkLayerType>IP</LinkLayerType>
              </Layer>
              <Layer>
                <Overhead>34</Overhead>
              </Layer>
              <Layer>
                <Overhead>16</Overhead>
                <UnitSize>53</UnitSize>
                <UnitOverhead>5</UnitOverhead>
              </Layer>
              <Layer>
                <LinkLayerType>Sonet</LinkLayerType>
              </Layer>
            </LayerInfo>
          </LogicalNetworkLink>
        </Parameters>
      </addNetworkZone>
    </Methods>
  </Command>

```

Figure 99 Add Network Zone (Composite type) request:

```

<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>|TransIf</Interface>
    <Methods>
      <addNetworkZone usn="1" version="1.0">
        <Parameters>
          <ID>236</ID>
          <Name>router3</Name>
          <Service>NAT</Service>
          <Parent>
            <ParentID>234</ParentID>
          </Parent>
          <IntraZoneBWInfo>
            <BWCIR>321</BWCIR>
            <NNSC>
              <Premium>12</Premium>
              <Platinum>23</Platinum>
              <Gold>25</Gold>
              <Silver>3</Silver>
              <Bronze>5</Bronze>
            </NNSC>
          </IntraZoneBWInfo>
          <LogicalNetworkLink>
            <BandWidthInfo>
              <Symmetry>>false</Symmetry>
              <UpBWInfo>
                <BWCIR>456</BWCIR>
              </UpBWInfo>
              <DownBWInfo>
                <BWCIR>997</BWCIR>
              </DownBWInfo>
            </BandWidthInfo>
            <LayerInfo>
              <Layer>
                <LinkLayerType>IP</LinkLayerType>
              </Layer>
              <Layer>
                <Overhead>34</Overhead>
              </Layer>
              <Layer>
                <Overhead>16</Overhead>
                <UnitSize>53</UnitSize>
                <UnitOverhead>5</UnitOverhead>
              </Layer>
              <Layer>
                <LinkLayerType>Sonet</LinkLayerType>
              </Layer>
            </LayerInfo>
          </LogicalNetworkLink>
        </Parameters>
      </addNetworkZone>
    </Methods>
  </Command>

```

Figure 100An example of Add Network Zone response:

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <addNetworkZone usn="1" version="1.0">
        <ReturnData>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </addNetworkZone>
    </Methods>
  </Response>
</CommandList>
```

Note: The node provisioning application will ignore the details of the Logical Network Link information since it does not need it. This information is required by the Session Policy Controller. If details of Logical Network Link information are included, Network VCAC must be on for the command to succeed.

Note: At least one of the tags Service and LogicalNetworkLink must be present. If both Service and LogicalNetworkLink tags are present, the Zone is a composite NAT & LBL zone.

6.7.29 Change Network Zone Command

A Network zone provisioned within the CS2000 management system can be modified such that a new parent network zone is selected.

This is a subset of the functionality supported by the interface and available to the SPC platform. Attempts to modify parameters other than the parent network zone will either be ignored - or will result in an error being returned. An error indicates that the change request would result in a mismatch between the CS2000 management platform and the SPC.

Two examples of a Change Network Zone request are shown, the first is valid for NAT, LBL and Composite type network zones, the second is only valid for LBL and Composite type network zones (if issued against a NAT it implies an unsupported type change from NAT to Composite). Both examples perform the same task as only the parent network zone change is of interest.

Figure 101 An example of Change Network Zone (Any Network Zone type) request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <changeNetworkZone usn="1" version="1.0">
        <Parameters>
          <Name>router1</Name>
          <Parent>
            <ParentID>247</ParentID>
          </Parent>
        </Parameters>
      </changeNetworkZone>
    </Methods>
  </Command>
</CommandList>
```

Figure 102 An example of Change Network Zone (LBL or Composite) request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <changeNetworkZone usn="1" version="1.0">
        <Parameters>
          <Name>router1</Name>
          <Parent>
            <ParentID>247</ParentID>
          </Parent>
          <IntraZoneBWInfo>
            <BWCIR>362</BWCIR>
            <NNSC>
              <Premium>12</Premium>
              <Platinum>23</Platinum>
              <Gold>2</Gold>
              <Silver>3</Silver>
              <Bronze>5</Bronze>
            </NNSC>
          </IntraZoneBWInfo>
          <LogicalNetworkLink>
            <BandWidthInfo>
              <Symmetry>false</Symmetry>
              <UpBWInfo>
                <BWCIR>456</BWCIR>
              </UpBWInfo>
              <DownBWInfo>
                <BWCIR>997</BWCIR>
              </DownBWInfo>
            </BandWidthInfo>
            <LayerInfo>
              <Layer>
                <LinkLayerType>IP</LinkLayerType>
              </Layer>
              <Layer>
                <Overhead>34</Overhead>
              </Layer>
              <Layer>
                <Overhead>16</Overhead>
                <UnitSize>53</UnitSize>
                <UnitOverhead>5</UnitOverhead>
              </Layer>
              <Layer>
                <LinkLayerType>Sonet</LinkLayerType>
              </Layer>
            </LayerInfo>
          </LogicalNetworkLink>
        </Parameters>
      </changeNetworkZone>
    </Methods>
  </Command>
</CommandList>
```


Figure 103 An example of Change Network Zone response:

```

<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <changeNetworkZone usn="1" version="1.0">
        <ReturnData>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </changeNetworkZone>
    </Methods>
  </Response>
</CommandList>

```

6.7.30 Delete Network Zone Command

Delete network zone removes a provisioned network zone from the CS2000 management system. The network zone to be removed may be identified by either its name or ID.

Figure 104 An example of Delete Network Zone request:

```

<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <deleteNetworkZone usn="1" version="1.0">
        <Parameters>
          <ID>352</ID>
        </Parameters>
      </deleteNetworkZone>
    </Methods>
  </Command>
</CommandList>

```

Figure 105 An example of Delete Network Zone response:

```

<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <deleteNetworkZone usn="1" version="1.0">
        <ReturnData>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </deleteNetworkZone>
    </Methods>
  </Response>
</CommandList>

```

6.7.31 Query Network Zone Command

Query network zone provides a mechanism to retrieve a single network zone, all network zones or a range of network zones restricted by ID filters.

Figure 106 An example of Query Network Zone (All zones) request:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryNetworkZone usn="1" version="1.0">
        <Parameters>
          </Parameters>
        </queryNetworkZone>
      </Methods>
    </Command>
  </CommandList>
```

Figure 107 An example of Query Network Zone (range of zones) request:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryNetworkZone usn="1" version="1.0">
        <Parameters>
          <IDMin>500</IDMin>
          <MaxZones>200</MaxZones>
        </Parameters>
      </queryNetworkZone>
    </Methods>
  </Command>
</CommandList>
```

Figure 108 An example of Query Network Zone (Specific zone) request:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryNetworkZone usn="1" version="1.0">
        <Parameters>
          <Name>router12</Name>
        </Parameters>
      </queryNetworkZone>
    </Methods>
  </Command>
</CommandList>
```

Figure 109 An example of Query Network Zone (range /all zones) response:

```
<?xml version="1.0"?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryNetworkZone usn="1" version="1.0">
        <ReturnData>
          <NetworkZone>
            <ID>235</ID>
            <Name>NZ1</Name>
            <ParentID>0</ParentID>
          </NetworkZone>
          <NetworkZone>
            <ID>236</ID>
            <Name>Peter8</Name>
            <ParentID>0</ParentID>
          </NetworkZone>
          <NetworkZone>
            <ID>566</ID>
            <Name>nat2</Name>
            <ParentID>65546</ParentID>
          </NetworkZone>
          <NetworkZone>
            <ID>891</ID>
            <Name>NZ2</Name>
            <ParentID>235</ParentID>
          </NetworkZone>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </queryNetworkZone>
    </Methods>
  </Response>
</CommandList>
```

Figure 110 An example of Query Network Zone (individual Composite network zone) response:

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryNetworkZone usn="1" version="1.0">
        <ReturnData>
          <ID>236</ID>
          <Name>Peter8</Name>
          <Service>NAT</Service>
          <Parent>
            <ParentID>0</ParentID>
          </Parent>
          <LogicalNetworkLink>
            <LayerInfo>
              <Layer>
                <LinkLayerType>UNKNOWN</LinkLayerType>
              </Layer>
            </LayerInfo>
          </LogicalNetworkLink>
          <ReturnCode value="0" text="Successful result" />
        </ReturnData>
      </queryNetworkZone>
    </Methods>
  </Response>
</CommandList>
```

6.7.32 Internet Transparency unsuccessful response messages

Figure 111 An example of Internet transparency error response

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>ITransIf</Interface>
    <Methods>
      <queryNAT usn="1" version="1.0">
        <ReturnData>
          <NATname>nat1</NATname>
          <ReturnCode value="305" text="NAT not found"/>
        </ReturnData>
      </queryNAT>
    </Methods>
  </Response>
</CommandList>
```

6.7.33 Limitations and restrictions

- *assocMG* (Associate Media Gateway)

- The profile applied to the MG must have a service type of ITRANS when a NAT or LBL middle box is specified.
 - The profile applied to the MG must have a service type of ITRANS_ROAM when root middleboxes are specified.
 - The GWC specified must have the Internet Transparency capability (ability to handle NAT/LBL and MPs) when a NAT or LBL middle box is specified.
 - The GWC specified cannot have more than 2000 middleboxes associated.
 - The NAT, LBL, or Root Middleboxes names passed to *assocMG* must exist.
 - 'outtsp' cannot be specified as a Root Middlebox.
 - The Pep server name tag cannot be used in a XML query containing the itransMiddlebox name tag.
 - The PepServer name tag or the itransMiddlebox name tag cannot be used in a XML query containing the rootMiddlebox name tags.
- queryMP (Query Media Proxy)
 - the MP name and GWC name must not be specified together in the same query.
- queryNAT
 - the NAT name and GWC name must not be specified together in the same query.
- changeAssocMB (Change NAT/LBL associated to a Media Gateway).
 - the specified MG must be defined.
 - the specified NAT/LBL middlebox must be defined.
- changeRootMiddleboxes (Change the root Middleboxes associated to a Media Gateway)
 - the specified MG must be defined

- the specified root middleboxes must be defined
 - 'outtsp' cannot be a specified middlebox
 - the specified MG must be defined
- addLBL
 - 'outtsp' cannot be specified as the parent of an LBL
- addNAT
 - 'outtsp' cannot be specified as the parent of a NAT
- addNetworkZone
 - specification of network zone ID requires Call Agent ID to be setup.
 - Network zones **should not be added** without the presence of the SPC platform and external VCAC counting being enabled.
- changeNetworkZone
 - changing network zone name is unsupported.
 - changing LogicalNetworkLink BandwidthInfo for a network zone of type NAT is unsupported (implies change of type from NAT to Composite).
 - deleting LogicalNetworkLink for a network zone is unsupported as implies potential change of type.
 - Removal of network zone parent must be done by setting ParentID = 0. Cannot perform using an empty ParentName tag.
- queryNetworkZone
 - LBL / Composite type network zone indicated by LogicalNetworkLink populated by dummy data.
- addLinkLayerType
 - Interface unsupported by CS2000 management system.
- changeLinkLayerType
 - Interface unsupported by CS2000 management system.

- deleteLinkLayerType
 - Interface unsupported by CS2000 management system.
- queryLinkLayerType
 - Interface unsupported by CS2000 management system.

6.8 Session Policy Controller (SPC) provisioning

The Session Policy controller (SPC) provide capability to apply policies during call-processing, e.g. policy-based codec selection, connection admission control etc. In general the policy may be implemented on the SPC itself.

In SN08, the SPC will support exactly one policy, that of Network Resource Reservation, using Virtual Connection Admission Control (VCAC) as the enforcement mechanism. Hence counting of available resource across Limited Bandwidth Links (LBLs) and connection admission decisions will be carried out on the SPC, instead of being done on the GWCs as in SN07. The SN08 GWCs communicate with the SPC to determine whether a call can be set-up. This will allow LBLs to be shared across different GWCs.

For provisioning, new XML interfaces are added for use in network installations where the CS2M platform is required to operate in conjunction with an SPC. These interfaces align with the provisioning model used by the SPC for NAT and VCAC functionality and is described in terms of Network Zones and Zone characteristics. The XML interfaces are identical to those supported by the SPC.

For XML commands supported from SPC interface refer Policy Controller Configuration Management NTP NN10432-511. In order to provision any entity, the information must first be provisioned using OSSGate node provisioning and then, if successful, the same information must be provisioned into the SPC. Node provisioning via OSSGate should be done first because all constraints (e.g. whether the zone is currently associated with a gateway) are verified. Thus if any constraints fail, the provisioning request will fail providing the proper response and will avoid SPC provisioning failures. It is important that the information is provisioned in the same order into each system. In particular, adding/deleting NATs & LBLs should be carried out identically in both systems. For example this could be achieved by adding an LBL to OSSGate node provisioning and then to SPC, followed by adding the next LBL to OSSGate node provisioning and then to the SPC etc.

The SN07 ITRANS VCAC interfaces for NAT and LBL provisioning are still available. However, when a SPC is deployed in the network configuration, the SPC consistent provisioning model is provided via use of the zone provisioning interfaces and the SN07 ITRANS VCAC interfaces for NAT and LBL provisioning are not required. A mapping is available in table below.

Need table that maps SN07 interface to SN08 interface.

Note: When adding a Network Zone, Resource Usage Factor and Max Count values are not required as long as Network VCAC is on. However if Network VCAC is 'off', resource usage factor and max count details ARE required and hence the XML command 'addLBL' should be used to add LBL Network Zones. However it is not recommend to switch Network VCAC off once it has been switched on. Also, if it is switched off, basic VCAC will only work correctly if the topology still meets the basic VCAC/'internal counting' topology constraints.

Add, delete, change of SPC are done from the CS2000 Management Tool GUI. For details refer GWC Configuration NTP NN10112-511.pdf

6.8.1 Limitations and Restrictions

- Only one SPC per CS2k is supported.
- Topology data transfer from CS2M node provisioning to SPC when SPC is commissioned has to be done manually using existing GUI/OSS interfaces.
- Once Network VCAC is switched on, CS2M node provisioning will remove any topology provisioning constraint checks which apply to Basic VCAC (internal counting). If Network VCAC is subsequently switched off, the internal counting checks will NOT be restored. It is not recommended to switch Network VCAC off once it has been switched on. Also, if it is switched off, basic VCAC will only work correctly if the topology still meets the basic VCAC/'internal counting' topology constraints. In normal use it is expected that Network VCAC will be switched on as soon as the upgrade and soak process is complete and SPC commissioned and will not be switched off.

6.9 Application Layer Gateway (ALG) provisioning

Application Layer gateways are logical interfaces that provide a way for network operators to separate the Cable hosted network IP address space from the service provider operator IP address space. It will allow the connection of two different IP address space domains without having to understand the exact IP address within the other domain.

ALGs are NATs that understand the application (in this case, NCS protocol) and enable the application to continue to work even though it is “behind” the NAT. The ALG provides a single IP address interface to the CS2K via a GWC while interfacing to multiple small gateways each with its own unique IP address. The IP address of the gateways which subtend the ALG are not known to the CS2K and thus will be set to 0.0.0.0 during provisioning. The ALG is positioned in the path of packets flowing from media gateways and media gateway controllers. In this level the ALG provides a level of packet translation without directly participating in the media flow, that is to say the ALG does not directly establish media path like a gateway, instead it serves as a proxy for the media through a request from the Call Server.

The OSS interface supports AssocMG, queryMG, ChangeMGGWCEMData, GetALGData and GetAlgNamesByGWCid.

AssocMG supports associating an ALG to a GW. The queryMG returns the associated ALG. The ChangeMGGWCEMData operation helps to change the ALG associated to a gateway to a different one. GetALGData returns the provisioning parameters associated with an ALG. GetAlgNamesByGWCid returns all ALG's provisioned on the specific GWC

6.9.1 Description of Method Parameters

Method parameters are defined below. Input data is mandatory except where indicated otherwise.

1. **AssocMG** - Method to associate a gateway to the CS2K. This method has the following NEW parameters: (only NEW ones described, see standard documentation for the other definitions).
 - Input data:

- usn
 - version
 - algName (Optional, Cannot be included as well as pepServerName) - defines the ALG (optional) for this gateway.
 - Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - ReturnCode - indicates via an integer value if the command has been successful or, if not, the error type and includes a brief textual message with further information.
2. **GetALGData**- Method that retrieves. This method has the following parameters:
- Input data:
 - usn
 - version
 - algName - defines the new ALG name
 - Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - ReturnCode - indicates via an integer value if the command has been successful or, if not, the error type and includes a brief textual message with further information.
 - Returns algname, IP address, Port and Protocol for the given ALG name.
3. **ChangeMGGWCEMData**- Method that changes the ALG for a given gateway. This method has the following parameters:
- Input data:
 - usn
 - version
 - MGname indicates the gateway that this change command is for.
 - algName - defines the new ALG name
 - Output data:
 - usn (value should be the same as the input)

- version (value should be the same as the input)
- ReturnCode - indicates via an integer value if the command has been successful or, if not, the error type and includes a brief textual message with further information.
- 4. **QueryMG-** Method that retrieves ALGs which are associated with input Gateway provided as parameter. This method has the following parameters:
 - _usn
 - version
 - MGname indicates the gateway that this change command is for.
 - ReturnCode - indicates via an integer value if the command has been successful or, if not, the error type and includes a brief textual message with further information.
 - Returns ALG middlebox name associated with MGname provided
- 5. **GetAlgNamesByGWCid-** Method that retrieves. This method has the following parameters:
 - Input data:
 - usn
 - version
 - Gateway controller name where data will be searched.
 - Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - ReturnCode - indicates via an integer value if the command has been successful or, if not, the error type and includes a brief textual message with further information.
 - Provides all of the ALGs which are provisioned on the given Gateway Controller

Figure 112An example of Associate MG with ALG request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>cs2kCfgMgrIf</Interface>
    <Methods>
      <assocMG usn="1" version="1.0">
        <Parameters>
          <mgUIName>MGXML.5</mgUIName>
          <mgProfileName>MOTOROLAMTA_2</mgProfileName>
          <mgIpAddr>0.0.0.0</mgIpAddr>
          <mgProtocolType>4</mgProtocolType>
          <mgProtocolVersion>1.0</mgProtocolVersion>
          <mgProtocolPort>2427</mgProtocolPort>
          <gwcUIName>GWC-113</gwcUIName>
          <reservedTerminations>2</reservedTerminations>
          <algName>ALG</algName>
        </Parameters>
      </assocMG>
    </Methods>
  </Command>
</CommandList>
```

Figure 113 An example of Query MG with ALG response:

```

<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
<Response>
<Interface>cs2kCfgMgrIf</Interface>
<Methods><queryMG usn="1" version="1.0">
<ReturnData>
<Row>
    <mgUIList>billrgw.1</mgUIList>
    <mgIpList>0.0.0.0</mgIpList>
    <callServerId>RTP7</callServerId>
    <gwcUIName>GWC-113</gwcUIName>
    <svcTypeList>0</svcTypeList>
    <nodeName>C411 00 1</nodeName>
    <engrEndPoints>2</engrEndPoints>
    <protType>ncsprotocol</protType>
    <protVersion>1.0</protVersion>
    <protPort>2427</protPort>
    <profileName>MOTOROLAMTA_2</profileName>
    <algName>ALG1</algName>
</Row><RC>0</RC><MsgTxt>Query of a Single MG
was successful</MsgTxt>
</ReturnData>
</queryMG>
</Methods>
</Response>
</CommandList>

```

Figure 114 An example of Change Associated ALG to an MG request

```

<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
<Command>
    <Interface>ALGOSSIf</Interface>
    <Methods>
        <changeMGGWCEMData usn="135" version="1.0" >
            <Parameters>
                <MGUIName>MG1</MGUIName>
                <algName>billr.alg1</algName>
            </Parameters>
        </changeMGGWCEMData>
    </Methods>
</Command>
</CommandList>

```

Figure 115 An example of Change Associated ALG to an MG response

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Response>
    <Interface>ALGOSSIf</Interface>
    <Methods>
      <changeMGGWCEMData usn="135" version="1.0">
        <ReturnCode value="0" text="Successful result" />
        <MsgTxt>MG ALG association changed successfully</MsgTxt>
      </changeMGGWCEMData>
    </Methods>
  </Response>
</CommandList>
```

Figure 116 An example of GetALG data request

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>ALGOSSIf</Interface>
    <Methods>
      <getALGData usn="1" version="1.0" >
        <Parameters>
          <algName>billr.alg1</algName>
        </Parameters>
      </getALGData >
    </Methods>
  </Command>
</CommandList>
```

Figure 117 An example of GetALG data response

```

<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Response>
    <Interface>ALGOSSIf</Interface>
    <Methods>
      <getALGData usn=123 version=1.0>
        <ReturnCode value="0" text="Successful result" />
        <MsgTxt>Data Available</MsgTxt>
        <algName>billr.alg1</algName>
        <ServerType>DQoS</ServerType>
        <IPAddress>47.192.234.88</IPAddress>
        <Port>2427</Port>
      </getALGData>
      <getALGData usn=124 version=1.0>
        <ReturnCode value="0" text="Successful result" />
        <MsgTxt>Data Available</MsgTxt>
        <algName>billr.alg1</algName>
        <ServerType>DQoS</ServerType>
        <IPAddress>47.192.234.100</IPAddress>
        <Port>2427</Port>
      </getALGData>
    </Methods>
  </Response>
</CommandList>

```

Figure 118 An example of Query GWC to ALG Connections request

```

<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>ALGOSSIf</Interface>
    <Methods>
      <getAlgNamesbyGWCIId usn="1", version="1.0">
        <GWC>GWC-1</GWC>
      </getAlgNamesbyGWCIId>
      <getAlgNamesbyGWCIId usn="1", version="1.0">
        <GWC>GWC-5</GWC>
      </getAlgNamesbyGWCIId>
    </Methods>
  </Command>
</CommandList>

```


Figure 119 An example of Query GWC to ALG Connections response

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Response>
    <Interface>ALGOSSIf</Interface>
    <Methods>
      <getAlgNamesbyGWCIId usn="1" version="1.0">
        <ReturnCode value="0" text="Successful result" />
        <MsgTxt>Data Available</MsgTxt>
        <GWC>GWC_1</GWC>
        <algName>billr.alg1</algName>
        <algName>billr.alg2</algName>
        <algName>billr.alg3</algName>
        <algName>billr.alg4</algName>
        <algName>billr.alg5</algName>
      </getAlgNamesbyGWCIId>
      <getAlgNamesbyGWCIId usn="2" version="1.0">
        <ReturnCode value="0" text="Successful result" />
        <MsgTxt>Data Available</MsgTxt>
        <GWC>GWC_5</GWC>
        <algName>billr.alg6</algName>
        <algName>billr.alg7</algName>
        <algName>billr.alg8</algName>
      </getAlgNamesbyGWCIId>
    </Methods>
  </Response>
</CommandList>
```

6.9.2 Limitations and Restrictions

- The CS2K has a limit of 2000 “middleboxes” which is the number of ALGs, PEP servers, NATs and media Proxies combined.
- The CS2K has a limit of 200 ALG middlebox.
- A single GWC on the CS2K has a limit of 20 ALG middleboxes.
- ALGs can only be provision against MTA gateways.
- ALG only supports the NCS protocol.
- If the ALG does not transmit Audit Endpoint (AUEP) messages to the MTA before the MTA sends its first message (typically an RSIP) then the lines in the CS2K will appear SYSB because the Audit Endpoint is rejected or not responded to. This could happen in the event that the ALG loses all state information and requires the first MTA originated message to re-establish its bindings. To correct this problem, the MTA needs to send any

message, e.g. an RSIP or an off hook and this will re-establish the binding and clear the SYSB state on the line. This problem will not occur if the ALG vendor supports discovery through the use of DNS of the MTA IP Address before the MTA sends its first message.

- DQoS is not supported on an MTA behind an ALG. Nortel is not aware of any ALG on the market that supports DQOS.
- IPSEC is not supported since a NAT device will cause end to end (GWC to MTA) security to fail because of the IP address and message content manipulation done by the ALG.
- Packetcable Electronic Surveillance can not be fully supported with an ALG. This is because the specification requires the transport of the MTA IP address which is not available to the CS2K. However, regular DMS packet electronic surveillance is supported.
- If a Redirecting Media Gateway Controller (RMGC) is configured, the DNS servers must be configured correctly. For example the ALG upon receiving the response from the RMGC on the CS2K, must be able allow the gateway to resolve the Fully Qualified Domain Name(FQDN) in the message through DNS queries.
- DNS is not supported by the CS2K to discover the ALG IP address. We only support static configuration of the IP address of the ALG since it is not expected that there will be many ALGs.
- When an ALG is associated to a gateway a DQOS PEP Servers, Low Band Width(LBL), IP-VPN Nats will not be allowed on the same gateway.
- The IP address of the gateway being associated with an ALG will not be modified. If the IP address field is left blank, the default value of 0.0.0.0 will be provision against the gateway.
- CS2K QoS collector will collect statistics as reported by the ALG. There is no separation of statistics between the ALG boundaries.

6.9.3 Authorization for ALG Commands

The security for the GUI Client and OSSgate interface includes support for permission or 'Authorization' levels for commands. Each command is associated with one or more

user groups. In order to execute a command, a user must belong to at least one of the associated user groups. The user groups associated with the new /modified commands associated with this feature are specified in the table below.

Table 2 ALG Commands Authorization

Command	User Group				
	mgcadm	mgciprov	mgcmte	mgcsprov	mgcro
addALG	X	X			
deleteALG	X	X			
changeALG	X	X			
AssocMG	X	X			
queryMG	X	X	X	X	X
GetALGData	X	X	X	X	X
changeMGGWCEMData	X	X			
getAlgNamesByGWCid	X	X	X	X	X

6.9.4 ALG Return Codes

Table 8: Application Layer Gateway return codes

Return code	Meaning
0	Successful operation
2	Associate MG Operation invalid input from client interface
3	Incorrect command version
4	The GWC has no ALG connections
11	Gateway does not exists in Database

6.10 User Authorization for Internet Transparency operations

The security for the OSSGate interface includes support for permission or 'Authorization' levels for commands. Each operation is associated with one or more authorization groups. To execute a command, a user must belong to at least one of the associated authorization groups. The authorization groups associated with each Internet Transparency operation is listed in Table-8. (For assocMG and queryMG authorization levels, see Nodes Provisioning permissions).

Table-8 Internet Transparency Command Authorization Levels

Command	User Group				
	mgcadm	mgcrw	mgcmte	mgcsprov	mgcro
queryCallAgentId	X	X	X	X	X
setCallAgentId	X	X			
addNAT	X	X			
deleteNAT	X	X			
changeNAT	X	X			
queryNAT	X	X	X	X	X
addLBL	X	X			
deleteLBL	X	X			
changeLBL	X	X			
queryLBL	X	X	X	X	X
addRU	X	X			
deleteRU	X	X			
changeRU	X	X			
queryRU	X	X	X	X	X
addMP	X	X			
deleteMP	X	X			
changeMP	X	X			
queryMP	X	X	X	X	X

Table-8 Internet Transparency Command Authorization Levels

Command	User Group				
	mgcadm	mgcrw	mgcmte	mgcsprov	mgcro
changeAssocMB	X	X			
changeRootMiddleboxes	X	X			
addNetworkZone	X	X			
changeNetworkZone	X	X			
deleteNetworkZone	X	X			
queryNetworkZone	X	X	X	X	X

6.10.1 Internet Transparency Return Codes

Table 3 Internet Transparency return codes

Return code	Meaning
0	Successful operation
16	Associate MG Operation invalid input from client interface. Example A service type of ITRANS is required
21	Associate MG Operation failed to associate the MG with a GWC. Example Reason the number of distinct NAT's used exceeds the limit of 400 per GWC
301	Incorrect command version in Internet Transparency change/query request
302	Platform applications internal error: OSS Interface applications cannot connect to required server application
303	Error in the input data
304	GWC-Element Manger error during ITRANS operation
305	No DATA Available example: No NAT /MP found matching this query or query of a non-existing GWC
306	SPC Interface Unsupported By Platform Interface is valid according to XSD but unsupported by the CS2000 management system. Returned in response to addLinkLayerType, changeLinkLayerType, deleteLinkLayerType and queryLinkLayerType interfaces being called.
501	User is not a member of any of the user groups allowed to perform this action

6.11 H.323 Limitation and Restrictions

The following are current restrictions with respect to H.323 gateway provisioning.

Change gateway operation: Following modifications are not allowed

- Changing H.323 version. (This is set to “4.0”).
- Changing protocol from H.323 to another protocol.
- Changing from another protocol to H.323.

Change gateway controller operation GWCs can not be changed from being non-H.323 to being H.323. Likewise, they can not be changed from H.323 to non-H.323.

6.12 Querying Gateway Profiles

Starting in SN08, OSSGate supports query functionality to enable the OSS to query aspects of a GW profiles (profile certificates). The supported commands are

- **Get Certificate List** - This command returns all the valid profiles names that is currently known to the CS2K management tool system. Each profile name is listed tag-limited with <certificate>.
- **Get Certificate Data** - This command returns the details pertaining to the provisioned profile. The user inputs in the the name of the Profile Name with which the certificate was initially created.

Figure 120 An example of Get Certificate List request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList >
  <Command>
    <Interface>CertificateOSSIf</Interface>
    <Methods>
      <getCertificateList usn="1" version="1.0">
        <Parameters>
          </Parameters>
        </getCertificateList>
      </Methods>
    </Command>
  </CommandList>
```


Figure 121 An example of Get Certificate List response:

```

<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>CertificateOSSIf</Interface>
    <Methods>
      <getCertificateList usn="1" version="1.0"><ReturnCode value="13.0" text="Successful
/><certificate>ARRIS_TOUCHTONE_NN01_4</certificate>
<certificate>ARRIS_TOUCHTONE_NN02_4</certificate>
<certificate>ASKEY_LINE_GW_12</certificate>
<certificate>ASKEY_LINE_GW_30</certificate>
<certificate>ASKEY_LINE_GW_4</certificate>
<certificate>AUDIOCODES</certificate><certificate>CICM</certificate>
<certificate>CISCO_2600</certificate><certificate>CISCO_3600</certificate>
<certificate>CISCO_AS5300</certificate><certificate>CISCO_H323_IOS</certificate>
<certificate>CVX1800_2688</certificate><certificate>CVX600_612</certificate>
<certificate>DOMAIN_NAME</certificate><certificate>H323_PROXY</certificate>
<certificate>MEDIATRIX_LINE_GW_24</certificate>
<certificate>MEDIATRIX_LINE_GW_4</certificate>
<certificate>MGCP_LINE_GW_1</certificate>
<certificate>MOTOROLAMTA_1</certificate>
<certificate>MOTOROLAMTA_2</certificate>
<certificate>MOTOROLAMTA_4</certificate>
<certificate>NORTEL_BCM</certificate>
<certificate>NUERA_GX_ASPEN</certificate>
<certificate>NUERA_GX_MEGACO</certificate>
<certificate>PVG15K_1000_ASPEN</certificate>
<certificate>PVG15K_1000_MEGACO</certificate>
<certificate>PVG15K_ASPEN</certificate>
<certificate>PVG15K_MEGACO</certificate>
<certificate>PVG15K_PARTIAL_ASPEN</certificate><certificate>AFC</certificate>
<certificate>PVG15K_PARTIAL_MEGACO</certificate>
<certificate>AMBIT_LINE_GW_16</certificate>
<certificate>PVG7K_ASPEN</certificate>
<certificate>PVG7K_MEGACO</certificate>
<certificate>PVG_APG_ASPEN</certificate>
<certificate>PVG_APG_MEGACO</certificate>
<certificate>PVG_APG_VSP3_ASPEN</certificate>
<certificate>PVG_APG_VSP3_MEGACO</certificate>
<certificate>PVG_VSP3_ASPEN</certificate>
<certificate>PVG_VSP3_MEGACO</certificate>
<certificate>SN06_CICM</certificate><certificate>SUCCESSION_1000</certificate>
<certificate>TGCP</certificate><certificate>TOUCHTONE_NN01_1</certificate>
<certificate>TOUCHTONE_NN01_2</certificate>
<certificate>TOUCHTONE_NN01_3</certificate>
<certificate>TOUCHTONE_NN01_4</certificate><certificate>UAS</certificate>
<certificate>UE9000MG</certificate><certificate>UE9000MG_ABI</certificate>
<certificate>UE9000MG_ABI_IP</certificate>
<certificate>UE9000MG_IP</certificate><certificate>WESTELL</certificate>
</Methods>
  </Response>
</CommandList>

```

Figure 122 An example of Get Certificate Data request:

```

<?xml version="1.0" encoding="UTF-8" ?>
<CommandList >
  <Command>
    <Interface>CertificateOSSIf</Interface>
    <Methods>
      <getCertificateData usn="1" version="1.0">
        <Parameters>
          <CertificateName>CISCO_2600</CertificateName>
        </Parameters>
      </getCertificateData>
    </Methods>
  </Command>
</CommandList>

```

Figure 123 An example of Get Certificate Data response

```

<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>CertificateOSSIf</Interface>
    <Methods>
      <getCertificateData usn="1" version="1.0"><ReturnCode value="13.0"
        text="Successful result" /><certificate> <MaxEndpoints>1032</MaxEndpoints>
        <Category>LARGE</Category>
        <EndpointType>nocarrierTrkGwcNN</EndpointType>
        <GenerateLGRP>true</GenerateLGRP>
        <ResvTermMandatory>true</ResvTermMandatory>
        <ChangeIPAvailable>true</ChangeIPAvailable>
        <DispPhyLocation>false</DispPhyLocation>
        <InventoryType>H.323 Gateway</InventoryType>
        <InventoryRole>Media Gateway</InventoryRole>
        <SupportedProtocol>
          <protocolName>h323</protocolName>
          <version>4.0</version>
        </SupportedProtocol>
        <GWCPProfileNumber>63</GWCPProfileNumber>
        <EPIDGenDesc>auto1</EPIDGenDesc>
        <ServiceTypeList>H323</ServiceTypeList>
        <ServiceTypeList>ITRANS</ServiceTypeList>
        <CompatibleGWProfileList>CISCO_2600</CompatibleGWProfileList>

```

Output continued

```

<GatewayNameFormatList>
  <nameFormat formatKey="GATEWAY.H323" enabled="true" applyToPattern=".*">
    <name>H323 Gateway</name>
    <delimiters>'</delimiters>
    <maxTokens>1</maxTokens>
    <tokenFormats>
      <tokenFormat tokenKey="GW">
        <name>Name</name>
        <pattern><![CDATA[[^"'<=>=]*]]></pattern>
      </tokenFormat>
    </tokenFormats>
  </nameFormat>
  <nameFormat formatKey="GATEWAY.ALL" enabled="true" applyToPattern=".*">
    <name>Generic For All Gateways</name>
    <template>gatewayName</template>
    <maxLength>32</maxLength>
    <minLength>1</minLength>
    <delimiters>none</delimiters>
    <tokenFormats>
      <tokenFormat tokenKey="GW">
        <pattern><![CDATA[[^"'<=>=,;]*]]></pattern>
        <name>Name</name>
      </tokenFormat>
    </tokenFormats>
  </nameFormat>
</GatewayNameFormatList>
<EndpointNameFormatList>
  <nameFormat formatKey="CARRIER.H323.EPG" enabled="true" applyToPattern=".*">
    <name>EPG</name>
    <template><![CDATA[EPG_<nnn>]]></template>
    <maxTokens>2</maxTokens>
    <minTokens>2</minTokens>
    <delimiters>_</delimiters>
    <maxLength>7</maxLength>
    <minLength>7</minLength>
    <appDataList>
      <appData name="CARRIERTYPEORPROTOCOLALLOWED" value="H323"
        type="NONE" />
      <appData name="CAPACITYDEFAULT" value="32" type="INTL" />
      <appData name="CAPACITYDEFAULT" value="24" type="NA" />
    </appDataList>
    <tokenFormats>
      <tokenFormat tokenKey="TYPE">
        <name>Type</name>
        <template>EPG</template>
        <type>alphanumeric</type>
        <values>EPG</values>
      </tokenFormat>
      <tokenFormat tokenKey="NUMBER">
        <minValue>0</minValue>
        <maxValue>999</maxValue>
        <name>Number</name>
        <type>numeric</type>
        <template><![CDATA[<nnn>]]></template>
      </tokenFormat>
    </tokenFormats>
  </nameFormat>
</EndpointNameFormatList>
</certificate></getCertificateData>    </Methods>
</Response>
</CommandList>

```


7: Provisioning Third party large line gateway and end-points.

Third party large line gateway provisioning is supported from SN08 from OSSGate. To provision these gateways the gateway profiles need to be known to the CS2K management tool system. Refer to details in GWC Configuration NTP NN10112-511.pdf for new profile addition. A typical sequence to add a third party gateway, provision configuration data and bring the system to service includes:

- Provision table SITE in CS2K Core. SITE values for third party gateways should be restricted for use for this type of gateways.
- Provision table PADDDATA in CS2K Core (PKLNL and PKNIL entries).
- Add the gateway profile certificate to the CS2K management tool system.
- Provision the third party gateway using CS2K management tool or OSSGate node provisioning interface
- Pre-provision the GW end-points from OSSGATE or BPT.
- Execute SERVORD+ commands via OSSGate when service is assigned to LENS in LNINV.

7.1 Associating Third party Gateways to GWC

Existing AssociateMG XML interface supports associating third party gateways via OSSGate. Additional XML elements that are applicable to the associateMG operation are frameNumber, unitNumber, frameType, floorPosition, rowPosition, framePosition, unitPosition, slotPosition. This information enables the Gateway and End-point Name mapping to Logical Group and Line Equipment Number.

7.1.1 End-point Name Mapping to Logical Group and Line Equipment Number (LEN)

The logical group name is derived from the Gateway SITE, the frame number and the unit number. The end-point name is user defined and its associated LEN has absolutely no bearing on the end-point name. The

LEN is derived from the logical group name and the user specified circuit number.

Example.

Gateway <GWNAME>, e.g. "MyGW"

Logical Group <SITE> <FFF> <U>, e.g. "CARY 142 0"

where:

<SITE> = SITE name chosen at gateway provisioning time.

Matching gateway name to SITE name is not an enforced rule, however it is suggested that a third party GW SITE name be defined in the CS2K Core and that all third party gateway provisioning utilize the "third party GW" SITE name.

<FFF> = Frame number that is specified during the third party GW provisioning. This number can range between 0 - 511

<U> = Unit number that is specified during the third party GW provisioning. This number can range from 0 - 9.

End-point

The end-point name is flexible. The format of the name is defined during the GW profile creation time and enforced during the GW provisioning time. The name is completely user defined. The name can include any physical location data of the hardware. The end-point name can be up to 32 characters long with no spaces in between.

LEN Format

<SITE> <FFF> <U> <TT> <tt>, e.g. "CARY 142 0 10 13"

where:

<SITE>, <FFF>, <U>, <TT>, and <tt> are as defined above.

The following summarizes the correlation between a GW, its end-points and the associated LENs:

- One third party "Gateway" mapped to one logical group in the CS2K Core.
- 1023 terminations are supported per logical group.

- 1 terminal number is reserved in the CS2K Core for maintenance/messaging.

Table 9: GW, LGRP, End-point, TN and LEN Relationship

GW Name (User Input)	Profile	LGRP Name	End-point (User Input)	Circuit Number (User Input)	LEN
MyGW.ext.net	PRF_1	CARY 100 5	ext/100/5/1/1	0	EXT 100 5 00 00
MyGW.ext.net	PRF_1	CARY 100 5	ext/100/5/1/2	1	EXT 100 5 00 01
:		:	:	:	:
:		:	:	:	:
MyGW2.ext.net	PRF_2	EXT2 200 8	ext2-ep-last	1022	EXT2 200 8 10 22

Terminal number 0 is reserved in the CS2K Core for the maintenance/messaging terminal on the node.

LEN is generated using the following formula:

(LGRPName) (CN/100) (CN%100) where CN => Circuit Number

The last 2 parts of LEN are 0 padded to keep the value to 2 digits similar to the other LENS.

7.2 Terminology and Description of Method Parameters

Chapter 6 details the various parameters needed for Associate MG operation. This sections details the additional parameters needed for AssociateMG operation to successfully provision a third party gateway.

frameNumber - Frame number. The range of this value is 0 - 511.
TYPE *unsignedShort*.

unitNumber - Unit/Shelf number. The range of this value is 0 - 9.
TYPE *unsignedShort*.

frameType - Frame Type. These types are pre-defined as the data type FRAME_NAME in the Core. TYPE *string*.

Values in the following list are valid:

{TME,DCE,RSE,N6E,CCC,MTN,NET,DNI,PDC,SLC,MIS,MSS,MEX,LGE,DTE,LTE

,LCE,MS6E,MS7E,ST6E,SPR,RTR,ST7E,PCSM,IOE,RTS,RUB,SME,RCE,RLCM,ID
TE,DME,OPE,RCME,SCE,DLE,DNPC,PCMM,PCPM,DICP,IAE,ISTE,CMDC,DPCC,
MSSC,MSDC,TAE,RUC,MCCM,MCNM,MCGM,MCLM,MCTM,MCPM,MCAM,M
CSM,MCMM,MCDM,MCRM,MCHM,MCBM,LIM,LRE,VSRE,ILGE,LTEI,PCLM,P
CTM,PCAM,TACE,THE,LCEI,MCDR,MC7M,MCEP,MCEM,MCEX,MCAP,MCAA,
RCEI,MCOR,MLNK,MC6M,MNET,IPEC,SCC,MCRM,MCLME,MCLMI,MCTMI,
MCAM2,EMC,DTEI,CRSC,CEXT,PMTC,PMPM,ICGM,CTME,CIPE,CIDE,CMS7,CI
OE,CDSN,HSI,MCRMS,ISME,CLCE,CMTA,CMVI,MVIE,CGPP,LGCI,SMUI,SPME,
RSCM,CDTO,TRLE,LGEI,UR256,UR512,UR1152,VLCE,CDTE,CIDT,CLTE,CRLC,
CISM,CPDT,CLMI,CPDC,CMIS,CMSS,CSDM,CCPE,MTAE,OPM256,SCPM,CSME,
MCGS,UEE,RMME,RSCE,SMGE,HUB,MCNI,MIEC,CDT2,DTEO,MVDD,MG9F,O
THR,DDRM}

floorPosition - Floor position. The range of this value is 0 - 99.
TYPE *unsignedShort*.

rowPosition - Row position. This value is pre-defined in the core.
TYPE *string*.

Values in the following list are

{A,B,C,D,E,F,G,H,I,J,K,L,M,N,P,Q,R,S,T,U,V,W,X,Y,Z,AA,BB,CC,DD,EE,FF,GG,HH,II,
JJ,KK,LL,MM,NN,PP,QQ,RR,SS,TT,UU,VV,WW,XX,YY,ZZ}

framePosition - Frame position. The range of this value is 0 - 99.
TYPE *unsignedShort*.

unitPosition - Unit/Shelf position. The range of this value is 0 - 99.
TYPE *unsignedShort*.

slotposition - Slot position. TYPE *unsignedShort*

Figure 124 An example of Associate MG request for a third party gateway:

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>cs2kCfgMgrIf</Interface>
    <Methods>
      <assocMG usn="1" version="1.0">
        <Parameters>
          <mgUIName>TPARTY.us.net</mgUIName>
          <mgProfileName>TPARTY</mgProfileName>
          <mgIpAddr>2.2.2.4</mgIpAddr>
          <mgProtocolType>4</mgProtocolType>
          <mgProtocolVersion>1.0</mgProtocolVersion>
          <mgProtocolPort>2944</mgProtocolPort>
          <mgSiteName>CARY</mgSiteName>
          <gwcUIName>GWC-6</gwcUIName>
          <reservedTerminations>1023</reservedTerminations>
          <frameNumber>4</frameNumber>
          <unitNumber>6</unitNumber>
          <frameType>OTHR</frameType>
          <floorPosition>10</floorPosition>
          <rowPosition>BB</rowPosition>
          <framePosition>20</framePosition>
          <unitPosition>30</unitPosition>
        </Parameters>
      </assocMG>
    </Methods>
  </Command>
</CommandList>
```

Figure 125 An example of Query MG response for a third pay gateway:

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
<Response>
<Interface>cs2kCfgMgrIf</Interface>
<Methods>
  <queryMG usn="1" version="1.0">
    <ReturnData>
      <Row>
        <mgUIList>TPARTY.us.net</mgUIList>
        <mgIpList>2.2.2.4</mgIpList>
        <callServerId>COMPACT5</callServerId>
        <gwcUIName>GWC-6</gwcUIName>
        <svcTypeList>0</svcTypeList>
        <nodeName>CARY 04 6</nodeName>
        <pepServerName>NOT_SET</pepServerName>
        <itransMiddleboxName>NOT_SET</itransMiddleboxName>
        <algName>NOT_SET</algName>
        <engrEndPoints>1023</engrEndPoints>
        <protType>megaco</protType>
        <protVersion>1.0</protVersion>
        <protPort>2944</protPort>
        <profileName>TPARTY</profileName>
        <maxEngrEndPoints>1023</maxEngrEndPoints>
        <rootMiddleboxName1 />
        <rootMiddleboxName2 />
        <rootMiddleboxName3 />
        <rootMiddleboxName4 />
        <rootMiddleboxName5 />
        <frameNumber>4</frameNumber>
        <unitNumber>6</unitNumber>
        <frameType>OTHR</frameType>
        <floorPos>10</floorPos>
        <rowPos>BB</rowPos>
        <framePos>20</framePos>
        <unitPos>30</unitPos>
      </Row>
      <RC>0</RC>
      <MsgTxt>Query of a Single MG was successful</MsgTxt>
    </ReturnData>
  </queryMG>
</Methods>
</Response>
</CommandList>
```

7.3 End-point provisioning

Third party gateway end-point pre-provisioning using XML is supported from OSSGATE interface. Following are the supported XML commands:

1. addLineEndpoint
2. delLineEndpoint
3. queryLineEndpoint

4. queryAllLineEndpoints

7.4 Terminology and Description of Method Parameters

- **mgUIName** - Fully qualified gateway name. This can be 1 - 32 characters long. TYPE *string*.
- **endpointName** - Name of the end-point that is being provisioned. This can be 1 - 32 characters long. TYPE *string*.
- **circuitNumber** - Number to associate the terminal with. The range of this value is 0 - 1022. TYPE *unsignedShort*.
- **cardCode** - String that defines the card code associated with the card. TYPE *string*. Optional parameter. The default is RDLSG.
- **padGroup** - String that defines the pad group. TYPE *string*. Optional parameter. The default is PKLNL.
- **ground** - String that defines the ground property. TYPE *string*. Values can be "N" or "Y". Optional parameter. The default is "N".
- **version** - The version refers to the version of the method or operation - e.g., the AddSubscriber operation may have version 1.0 and 2.0, where version 1.0 takes parameters X and Y and version 2.0 takes parms X, Y, and Z. TYPE *decimal*.

7.4.1 Description of Method Parameters

Method parameters are defined below. Input data is mandatory except where indicated otherwise.

addLineEndpoint - Interface/Method that adds a specified end-point to the SESM database and allocates a LEN to the corresponding terminal in CS2K Core.

- Input data:
 - usn
 - version
 - mgUIName
 - endpointName
 - cardCode - Default: RDTEBS

- padGroup - Default: STDLN
 - ground - Default: N
- Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - RC - Return Code
 - MsgTxt - Text data detailing specific results of the operation in terms of success or failure of the operation and possible indication of failure reason incase of a failure.

delLineEndpoint - Interface/Method that deletes a specified end-point from the SESM database and de-allocates LEN from the corresponding terminal in CS2K Core.

Pre-condition: Service on the end-point must be un-assigned before attempting this operation.

- Input data:
 - usn
 - version
 - mgUIName
 - endpointName

OR

To delete a bunch of end-points with a specified range of circuit numbers,

 - circuitNumberBegin
 - circuitNumberEnd

OR

To delete ALL the end-points on this GW

 - ALL
- Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - RC - Return Code

- MsgTxt - Text data detailing specific results of the operation in terms of success or failure of the operation and possible indication of failure reason incase of a failure.

queryLineEndpoint - Interface/Method that queries a specific end-point data on this gateway.

The end-point data will be enclosed within the <Row> tags similar to the output of the other nodes XML operations.

- Input data:
 - usn
 - version
 - mgUIName
 - endpointName
- Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - mgUIName
 - endpointName
 - CircuitNumber
 - cardCode
 - padGroup
 - status
 - ground
 - status
 - tid
 - LEN
 - RC - Return Code
 - MsgTxt - Text data detailing specific results of the operation in terms of success or failure of the operation and possible indication of failure reason incase of a failure.

queryAllLineEndpoints - Interface/Method that queries all the end-points on this gateway.

The end-point data for each of the end-points will be enclosed within the <Row> tags similar to the output of the other nodes XML operations.

- Input data:
 - usn
 - version
 - mgUIName
- Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - mgUIName
 - endpointName
 - CircuitNumber
 - cardCode
 - padGroup
 - status
 - ground
 - status
 - tid
 - LEN
 - RC - Return Code
 - MsgTxt - Text data detailing specific results of the operation in terms of success or failure of the operation and possible indication of failure reason incase of a failure.

Figure 126 An example of addLineEndpoint XML request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
<Command>
  <Interface>cs2kCfgMgrIf</Interface>
  <Methods>
    <addLineEndpoint usn="1" version="1.0">
      <Parameters>
        <mgUIName>TPARTY.us.net</mgUIName>
        <endpointName>aaln/1</endpointName>
        <circuitNumber>1</circuitNumber>
      </Parameters>
    </addLineEndpoint>
  </Methods>
</Command>
</CommandList>
```

Figure 127 An example of addLineEndpoint XML response:

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
<Response>
  <Interface>cs2kCfgMgrIf</Interface>
  <Methods>
    <addLineEndpoint usn="1" version="1.0">
      <ReturnData>
        <RC>0</RC>
        <MsgTxt>The add line endpoint operation was successful: Gateway =
TPARTY.us.net, Endpoint = aaln/1.</MsgTxt>
      </ReturnData>
    </addLineEndpoint>
  </Methods>
</Response>
</CommandList>
```

Figure 128 An example of delLineEndpoint XML request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>cs2kCfgMgrIf</Interface>
    <Methods>
      <delLineEndpoint usn="1" version="1.0">
        <Parameters>
          <mgUIName>TPARTY.us.net</mgUIName>
          <endpointName>aaln/1</endpointName>
        </Parameters>
      </delLineEndpoint>
    </Methods>
  </Command>
</CommandList>
```

Figure 129 An example of delLineEndpoint XML response:

```

<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
<Response>
<Interface>cs2kCfgMgrIf</Interface>
  <Methods>
    <delLineEndpoint usn="1" version="1.0">
      <ReturnData>
        <RC>0</RC>
        <MsgTxt>
          The delete line endpoint operation was successful: Gateway = TPARTY.us.net, Endpoint =
          aaln/1.
        </MsgTxt>
      </ReturnData>
    </delLineEndpoint>
  </Methods>
</Response>
</CommandList>

```

Figure 130 An example of delLineEndpoint for a range of data XML request:

Example request for deleting a range of end-points (with circuit numbers ranging from 1 - 4 with limits inclusive) from a gateway:

```

<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>cs2kCfgMgrIf</Interface>
    <Methods>
      <delLineEndpoint usn="1" version="1.0">
        <Parameters>
          <mgUIName>TPARTY.us.net</mgUIName>
          <circuitNumberBegin>1</circuitNumberBegin>
          <circuitNumberEnd>4</circuitNumberEnd>
        </Parameters>
      </delLineEndpoint>
    </Methods>
  </Command>
</CommandList>

```


Figure 131 An example of **delLineEndpoint** XML request that Deletes ALL the end-point of this gateway:

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>cs2kCfgMgrIf</Interface>
    <Methods>
      <delLineEndpoint usn="1" version="1.0">
        <Parameters>
          <mgUIName>TPARTY.us.net</mgUIName>
        </Parameters>
      </delLineEndpoint>
    </Methods>
  </Command>
</CommandList>
```

Figure 132 An example of **queryLineEndpoint** XML request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>cs2kCfgMgrIf</Interface>
    <Methods>
      <queryLineEndpoint usn="1" version="1.0">
        <Parameters>
          <mgUIName>TPARTY.us.net</mgUIName>
          <endpointName>aaln/5</endpointName>
        </Parameters>
      </queryLineEndpoint>
    </Methods>
  </Command>
</CommandList>
```

Figure 133 An example of **queryLineEndpoint** XML response

```

<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
<Response>
<Interface>cs2kCfgMgrIf</Interface>
<Methods>
  <queryLineEndpoint usn="1" version="1.0">
    <ReturnData>
      <mgUIName>TPARTY.us.net</mgUIName>
      <Row>
        <endpointName>aaln/5</endpointName>
        <circuitNumber>5</circuitNumber>
        <cardCode>RDTLSG</cardCode>
        <padGroup>PKLNL</padGroup>
        <ground>N</ground>
        <status>HASU</status>
        <terminalNumber>6</terminalNumber>
        <LEN>CARY 04 6 00 05</LEN>
      </Row>
      <RC>0</RC>
      <MsgTxt>Query line endpoints operation was successful: Gateway = TPARTY.us.net
Endpoint = aaln/5.</MsgTxt>
    </ReturnData>
  </queryLineEndpoint>
</Methods>
</Response>
</CommandList>

```

Figure 134 An example of **qryAllLineEndpoints** XML request

```

<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
<Command>
  <Interface>cs2kCfgMgrIf</Interface>
  <Methods>
    <queryAllLineEndpoints usn="1" version="1.0">
      <Parameters>
        <mgUIName>TPARTY.us.net</mgUIName>
      </Parameters>
    </queryAllLineEndpoints>
  </Methods>
</Command>
</CommandList>

```

Note: queryAllLineEndpoints query can potentially return a big list.

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
<Response>
<Interface>cs2kCfgMgrIf</Interface>
<Methods>
<queryAllLineEndpoints usn="1" version="1.0">
<ReturnData>
<mgUIName>TPARTY.us.net</mgUIName>
<Row>
<endpointName>aaln/0</endpointName>
<circuitNumber>0</circuitNumber>
<cardCode>RDTLSG</cardCode>
<padGroup>PKLNL</padGroup>
<ground>N</ground>
<status>HASU</status>
<terminalNumber>1</terminalNumber>
<LEN>CARY 04 6 00 00</LEN>
</Row>
<Row>
<endpointName>aaln/5</endpointName>
<circuitNumber>5</circuitNumber>
<cardCode>RDTLSG</cardCode>
<padGroup>PKLNL</padGroup>
<ground>N</ground>
<status>HASU</status>
<terminalNumber>6</terminalNumber>
<LEN>CARY 04 6 00 05</LEN>
</Row>
<Row>
<endpointName>aaln/6</endpointName>
<circuitNumber>6</circuitNumber>
<cardCode>RDTLSG</cardCode>
<padGroup>PKLNL</padGroup>
<ground>N</ground>
<status>HASU</status>
<terminalNumber>7</terminalNumber>
<LEN>CARY 04 6 00 06</LEN>
</Row>
- <Row>
<endpointName>aaln/7</endpointName>
<circuitNumber>7</circuitNumber>
<cardCode>RDTLSG</cardCode>
<padGroup>PKLNL</padGroup>
<ground>N</ground>
<status>HASU</status>
<terminalNumber>8</terminalNumber>
<LEN>CARY 04 6 00 07</LEN>
</Row>
<RC>0</RC>
<MsgTxt>Query line endpoints operation was successful: Gateway = TP
</ReturnData>
</queryAllLineEndpoints>
</Methods>
</Response>
</CommandList>
```

7.5 Batch provisioning

Line Endpoint provisioning can be done using Batch Provisioning Tool (BPT). Details of the usage is explained in Chapter 12.

7.6 User Authorization for Line End-point provisioning

The security for the OSSGate interface includes support for permission or 'Authorization' levels for commands. Each operation is associated with one or more authorization groups. To execute a command, a user must belong to at least one of the associated authorization groups. The authorization groups associated with each operation is listed in Table-8. (For assocMG and queryMG authorization levels, see Nodes Provisioning permissions).

Table-8 Line End-point Provisioning Command Authorization Levels

Command	User Group				
	mgcadm	mgcrw	mgcmte	mgcsprov	mgcro
addLineEndpoint	X	X			
delLineEndpoint	X	X			
qryLineEndpoint	X	X	X	X	X

7.7 Return Codes

The following table lists the possible return codes from the Line Endpoint provisioning operations.

Table 10: Line Endpoint Provisioning Return Codes

Return Code	Meaning
0	Successful operation
72	Add Line Endpoint operation, invalid input from client
73	Add Line Endpoint operation, unsupported GW type
74	Add Line Endpoint operation, not sufficient input
75	Add Line Endpoint operation, input value not in a correct format
76	Add Line Endpoint operation, failed to read the gateway data from the database
77	Add Line Endpoint operation, failed to add the endpoint to the GWC
78	Add Line Endpoint operation, specified endpoint already exists
79	Add Line Endpoint operation, have reached the maximum supported endpoint capacity on the GWC
80	Add Line Endpoint operation, failed to add the endpoint to the Core although the endpoint was added to the GWC successfully
81	Add Line Endpoint operation, GWC failed to add the end point
82	Add Line Endpoint operation, encountered an error after successfully adding the endpoint to the GWC
83	Delete Line Endpoint operation, invalid input from client
84	Delete Line Endpoint operation, not sufficient input
85	Delete Line Endpoint operation, input value not in a correct format
86	Delete Line Endpoint operation, failed to read the gateway data from the database
87	Delete Line Endpoint operation, unsupported GW type
88	Delete Line Endpoint operation, failed to delete the endpoint from its GWC
89	Delete Line Endpoint operation, failed to delete the endpoint from the Core
90	Delete Line Endpoint operation, failed to get the end point data from the database
91	Delete Line Endpoint operation, failed to retrieve the end point data from the database
92	Delete Line Endpoint operation, circuit number range is not valid
93	Delete Line Endpoint operation, delete operation was successful only partially due to failures in deleting the endpoints from the Core

Table 10: Line Endpoint Provisioning Return Codes

Return Code	Meaning
94	Delete Line Endpoint operation, delete operation was successful only partially due to failures in deleting the endpoints from GWC
95	Delete Line Endpoint operation, delete operation was successful only partially
100	Query Line Endpoint operation, invalid input from client
101	Query Line Endpoint operation, no endpoints are found in GWCEM
102	Query Line Endpoint operation, failed to retrieve the LEN data from the Core
103	Query Line Endpoint operation, failed to retrieve the endpoint data from the GWCEM

7.8 Limitations and Restrictions

- Conversion of existing third party large line GW data to SN08 format is not supported. To use the third party gateway provisioning interface, the existing GW, associated end-points, service must be de-provisioned and re-added in SN08. Both new and old format data will be allowed to co-exist in SN08, however the existing GW and the end-points will be managed in the same format as defined in the previous release.
- The capabilities can not be fully utilized until all the components of the OAMP are at the base SN08 load line up. There is a dependency on the new LGRPTYPE in the SN08 CS2K Core that will be used by the third party GW nodes in the Core. The GWC devices are required to be on the 08 release as well to support spare profiles.

8: Carrier Provisioning with OSSGate

8.1 Supported Carrier Provisioning Commands

The following existing Carrier Provisioning operations are supported:

- Add Carrier
- Delete Carrier
- Get Carrier
- Get Endpoint
- List All Carriers

8.2 XML Commands

AddCarrier: This interfaces will add carriers to a trunk gateway or a End Point Groups (EPG) of size 24 or 32 endpoints to a H.323 Gateway dependent upon market configuration. For H.323 gateways In order to provision a Gateway with multiple EPGs, this command should be called repeatedly with a unique EPG. This command will automatically grab the first available block of contiguous TID space TNs available on the GWC if the optional “FirstTN” tag is not used. The user can also specify the starting TID location. This option gives users the ability to assign EPs to specific TID locations if they desire. If a firstTN tag is included, the system will assign a contiguous group of 24 or 32 EPs starting at the given location. If any of the TNs in the range are unavailable the command will fail.

8.2.1 Description of Method Parameters

Method parameters are defined below. Input data is mandatory except where indicated otherwise.

1. **addCarrier** - Method to add a group of carriers to any trunk gateway including H.323 gateways. This method has the following parameters:

- Input data
 - usn - Unique Sequence Number. This parameter is introduced to facilitate sending multiple XML commands in one file. This parameter value will be used to match the responses with the requests. TYPE *integer*.

- version - The version refers to the version of the method or operation - Current version is 1.0
 - mgName - A string value starting with PVG followed by alphanumeric or underscore
 - carrierName - A string value of size range 1 to 32 characters that represents the various carrier types like DS3, E1, SM, ST
 - svcData - A optional parameter. This can be one of PRI service type, V5.2 service type, ISUP or H.323
A PRI service type has one parameter, IID - an integer value between 0 to 254
A v5.2 service type has three parameters, V5.2IID, V5.2LinkID and V5.2UALinkId. All these parameters are integers 1 to 4094 range.
 - firstTN - A optional parameter. An integer value between 1 to 4094
 - tidSize - A optional parameter. An integer value between 1 to 4094
- Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - firstTID - A integer value that is a concatenation of node number and terminal number
 - numberOfTids - An integer value that indicates the total number of allocated tids for the carrier.
 - ReturnCode - indicates via an integer value if the command has been successful or, if not, the error type and includes a brief textual message with further information.

Operation getCarrier - The same functionality is provided by new operation listAllCarriers when used with the FILTER_CARRIER option.

2. **getCarrier** - Method to get a group of carriers associated to any trunk gateway including H.323 gateways. This method has the following parameters:

- Input data:
 - usn - Unique Sequence Number. This parameter is introduced to facilitate sending multiple XML commands in one file. This parameter value will be used to match the responses with the requests. TYPE *integer*.
 - version - The version refers to the version of the method or operation - Current version is 1.0
 - mgName A string value starting with PVG followed by alphanumeric or underscore
 - carrierName A string value of size range 1 to 32 characters that represents the various carrier types like DS3, E1, SM, ST
- Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - gwcName A string value starting with GWC- followed by a number between 0 to 255
 - mgName A string value starting with PVG followed by alphanumeric or underscore
 - carrierName A string value of size range 1 to 32 characters that represents the various carrier types like DS3, E1, SM, ST
 - svcData - A optional parameter. This can be one of PRI service type, V5.2 service type, ISUP or H.323
A PRI service type has one parameter, IID - an integer value between 0 to 254
A v5.2 service type has three parameters, V5.2IID, V5.2LinkID and V5.2UALinkId. All these parameters are integers 1 to 4094 range.

- service - The type of service the carrier is associated with. It can be one of PRI, V5.2, ISUP or H.323
- numberOfTids - An integer value that indicates the total number of allocated tids for the carrier.
- ReturnCode - indicates via an integer value if the command has been successful or, if not, the error type and includes a brief textual message with further information.

3. **getEndPoint** - Method to get endpoints associated with carriers on any trunk gateway including H.323 gateways. This method has the following parameters:

- Input data:

- usn - Unique Sequence Number. This parameter is introduced to facilitate sending multiple XML commands in one file. This parameter value will be used to match the responses with the requests. TYPE *integer*.
- version - The version refers to the version of the method or operation - Current version is 1.0
- mgName A string value starting with PVG followed by alphanumeric or underscore
- endpointName A string value of size range 1 to 32 characters that represents the various carrier types like DS3, E1, SM, ST

- Output data:

- usn (value should be the same as the input)
- version (value should be the same as the input)
- mgName A string value starting with PVG followed by alphanumeric or underscore
- endpointName A string value of size range 1 to 32 characters that represents the various carrier types like DS3, E1, SM, ST
- tid - A integer value that represent the Terminal-ID associated with the channel.
- ReturnCode - indicates via an integer value if the command has been successful or, if not, the error type and includes a brief textual message with further information.

4. **deleteCarrier** - Method to delete a group of carriers to any trunk gateway including H.323 gateways. This method has the following parameters:

- Input data:

- usn - Unique Sequence Number. This parameter is introduced to facilitate sending multiple XML commands in one file. This parameter value will be used to match the responses with the requests. TYPE *integer*.
- version - The version refers to the version of the method or operation - Current version is 1.0
- mgName A string value starting with PVG followed by alphanumeric or underscore
- carrierName A string value of size range 1 to 32 characters that represents the various carrier types like DS3, E1, SM, ST
-

- Output data:

- usn (value should be the same as the input)
- version (value should be the same as the input)
- ReturnCode - indicates via an integer value if the command has been successful or, if not, the error type and includes a brief textual message with further information.
-

5. **listAllCarrier** - Method to list all carriers associated to any trunk gateway including H.323 gateways. This method has the following parameters:

- Input data:

- usn - Unique Sequence Number. This parameter is introduced to facilitate sending multiple XML commands in one file. This parameter value will be used to match the responses with the requests. TYPE *integer*.
- version - The version refers to the version of the method or operation - Current version is 1.0
- FILTER_GWC A option from which one of the following four may be specified: FILTER_ALL, FILTER_GWC, FILTER_MG, FILTER_CARRIER.

- **gwcname** A string value starting with GWC- followed by a number between 0 to 255
- Output data:
 - **usn** (value should be the same as the input)
 - **version** (value should be the same as the input)
 - **gwcname** A string value starting with GWC- followed by a number between 0 to 255
 - **mg name** A string value starting with PVG followed by alphanumeric or underscore
 - **carrier name** A string value of size range 1 to 32 characters that represents the various carrier types like DS3, E1, SM, ST
 - **svcData** This can be one of PRI, ISUP, V5.2, H.323. A PRI type has one parameter - IID an integer value between 1 to 4094
 - **service** - The type of service the carrier is associated with. It can be one of PRI, V5.2, ISUP or H.323
 - **IID** - Data associated with PRI service type. Integer value between 0 to 254
 - **endptList** - A list of all endpoints associated with the carrier. Each endpoint has a name (string) and associated tid value (a integer value that is a concatenation of node number and terminal number).
 - **ReturnCode** - indicates via an integer value if the command has been successful or, if not, the error type and includes a brief textual message with further information.

8.2.2 Examples of XML Command and Response Messages

Figure 135 An example of addCarrier request:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>EndptGrpProvIf</Interface>
    <Methods>
      <addCarrier usn="1" version="1.0">
        <Parameters>
          <mgName>PVG190</mgName>
          <carrierName>DS3_20.28</carrierName>
          <svcData><PRI>
            <IID>7</IID>
          </PRI>
        </svcData>
        <firstTN>649</firstTN>
      </Parameters>
    </addCarrier>
  </Methods>
</Command>
</CommandList>
```

Note: firstTN is an optional field.

Figure 136 An example of H.323 addCarrier request:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>EndptGrpProvIf</Interface>
    <Methods>
      <addCarrier usn="1" version="1.0">
        <Parameters>
          <mgName>PVG190</mgName>
          <carrierName>DS3_20.28</carrierName>
          <svcData> <H323 /> </svcData>
          <firstTN>649</firstTN>
        </Parameters>
      </addCarrier>
    </Methods>
  </Command>
</CommandList>
```

Figure 137 An example of addCarrier response:

```

<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
<Response>
<Interface>EndptGrpProvIf</Interface>
<Methods><addCarrier usn="1" version="1.0">
<ReturnData>
<Row><firstTID>13649</firstTID>
<numberOfTids>24</numberOfTids></Row>
<RC>0</RC>
<MsgTxt>
EGP:AddCarrierOpReplyStAdd CARRIER operation was successful
</MsgTxt>
</ReturnData>
</addCarrier>
</Methods>
</Response>
</CommandList>

```

Figure 138 An example of getCarrier request:

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CommandList >
<Command>
<Interface>EndptGrpProvIf</Interface>
<Methods>
<getCarrier usn="1" version="1.0">
<Parameters>
<mgName>PVG190</mgName>
<carrierName>DS3_20.1</carrierName>
</Parameters>
</getCarrier>
</Methods>
</Command>
</CommandList>

```

Figure 139 An example of getCarrier response:

```

<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Response><Interface>EndptGrpProvIf</Interface>
    <Methods>
      <getCarrier usn="1" version="1.0">
        <ReturnData>
          <gwcName>GWC-1</gwcName>
          <mgName>PVG190</mgName>
          <Row><carrierName>DS3_20.1</carrierName>
            <svcData><service>ISUP</service></svcData>
          </Row><RC>0</RC>
          <MsgTxt>Carrier was successfully retrieved</MsgTxt>
        </ReturnData>
      </getCarrier>
    </Methods>
  </Response>
</CommandList>

```

Figure 140 An example of getEndPoint request:

```

<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>EndptGrpProvIf</Interface>
    <Methods>
      <getEndPoint usn="1" version="1.0">
        <Parameters>
          <mgName>PVG8</mgName>
          <endpointName>DS3_20.1.23</endpointName>
        </Parameters>
      </getEndPoint>
    </Methods>
  </Command>
</CommandList>

```

Figure 141 An example of getEndPoint response:

```

<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Response>
    <Interface>EndptGrpProvIf</Interface>
    <Methods>
      <getEndPoint usn="1" version="1.0">
        <ReturnData>
          <mgName>PVG8</mgName>
          <endpointName>DS3_20.1.23</endpointName>
          <tid>200 23</tid>
          <RC>0</RC>
          <MsgTxt>Get Endpoint operation was successful.</MsgTxt>
        </ReturnData>
      </getEndPoint>
    </Methods>
  </Response>
</CommandList>

```

Operation deleteCarrier

Figure 142 An example of deleteCarrier request:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>EndptGrpProvIf</Interface>
    <Methods>
      <deleteCarrier usn="1" version="1.0">
        <Parameters>
          <mgName>PVG8</mgName>
          <carrierName>DS3_20.1</carrierName>
        </Parameters>
      </deleteCarrier>
    </Methods>
  </Command>
</CommandList>
```

Figure 143 An example of deleteCarrier response:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Response>
    <Interface>EndptGrpProvIf</Interface>
    <Methods>
      <deleteCarrier usn="1" version="1.0">
        <ReturnData>
          <RC>0</RC>
          <MsgTxt>Delete Carrier operation was successful.</MsgTxt>
        </ReturnData>
      </deleteCarrier>
    </Methods>
  </Response>
</CommandList>
```

Operation ListAllCarriers

Figure 144 An example of ListAllCarriers command request:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>EndptGrpProvIf</Interface>
    <Methods>
      <listAllCarriers usn="1" version="1.0">
        <Parameters>
          <FILTER_GWC>
            <gwcName>GWC-6</gwcName>
          </FILTER_GWC>
        </Parameters>
      </listAllCarriers>
    </Methods>
  </Command>
</CommandList>
```


Figure 145 An example of ListAllCarriers response:

```

<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
<Response>
<Interface>EndptGrpProvIf</Interface>
<Methods>
<listAllCarriers usn="1" version="1.0">
<ReturnData>
<gwc name="GWC-6">
    <mg name="TrunkMG">
<carrier name="DS3_20.2">
<svcData>
<service>PRI</service>
<IID>7</IID>
</svcData>
<endptList>
<endPt name="DS3_20.2.1"><tid>36 1</tid></endPt>
<endPt name="DS3_20.2.2"><tid>36 2</tid></endPt>
.
.
<endPt name="DS3_20.2.22"><tid>36 22</tid></endPt>
<endPt name="DS3_20.2.23"><tid>36 23</tid></endPt>
<endPt name="DS3_20.2.24"><tid>36 24</tid></endPt>
</endptList>
</carrier>
<carrier name="DS3_20.9">
<svcData>
<service>ISUP</service>
</svcData>
<endptList>
<endPt name="DS3_20.9.1"><tid>36 1401</tid></endPt>
<endPt name="DS3_20.9.2"><tid>36 1402</tid></endPt>
<endPt name="DS3_20.9.3"><tid>36 1403</tid></endPt>
.
.
<endPt name="DS3_20.9.22"><tid>36 1422</tid></endPt>
<endPt name="DS3_20.9.23"><tid>36 1423</tid></endPt>
<endPt name="DS3_20.9.24"><tid>36 1424</tid></endPt>
</endptList>
</carrier>
</mg>
</gwc>
<RC>0</RC>
<MsgTxt>
Retrieval of Carriers Successfully Completed. </MsgTxt>
</ReturnData>
</listAllCarriers>
</Methods>
</Response>
</CommandList>

```

8.3 User Authorization for Carrier Operations

The security for the OSSGate interface includes support for permission or 'Authorization' levels for commands. Each operation is associated with one or more user groups. In order to execute a command, a user must belong to at least one of the associated user groups. The user groups associated with each Carrier Provisioning command are specified in the Table-8.

Table 11: Carrier Provisioning user groups

Command	User Group				
	trkadm	trkrw	trksprov	trkmtc	trlro
addCarrier	X	X			
deleteCarrier	X	X			
getEndpoint	X	X	X	X	X
get Carrier	X	X	X	X	X
get CarrierByFilter	X	X	X	X	X

8.3.1 Return Codes

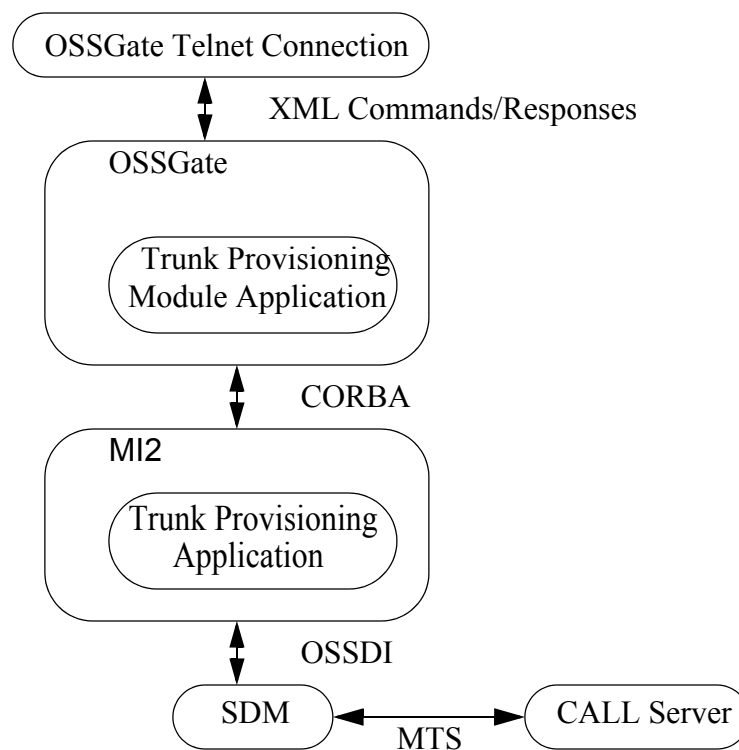
Table 12: Carrier Provisioning return codes

Return Code	Meaning
0	Successful operation
1	More data coming
2	Request Rejected due to Resource Limits Reached
3	Application was Commanded to Abort
4	Invalid input from client
5	Failed at GWC EM
6	Failed at Network View
7	Successful operation, but no data.
8	Timeout at GWC EM

9: Trunk Provisioning with OSSGate

The Trunk Provisioning application enables authenticated OSS clients to perform provisioning operations on legacy and MG-based trunk data from CS2000 Management tool server. The provisioning operations include adding, changing, listing and deleting tuples. Access to this functionality is by connection to OSSGate.

Figure 146 Provisioning in the Succession Network



Succession trunk provisioning of gateway-hosted endpoints is a multiple-step process:

1. Carrier endpoints on a MG must be provisioned by using the Bulk Carrier Provisioning application. The Carrier Provisioning application is responsible for allocating TIDs for all channels of a carrier.
2. Provision the CS2000 tables CLLI and TRKGRP with the appropriate data.

3. Provision trunk service on MG-based endpoints by using the Trunk Provisioning application to populate tables TRKSGRP and TRKMEM. The endpoint/TID which was allocated for the carrier channel in step 1 is specified when provisioning tables TRKSGRP and TRKMEM.

TRKSGRP tuples for PRI (Primary Rate Interface) trunk groups specify the D channel endpoint/TID. TRKMEM tuples specify endpoint/TID of a respective trunk group.

Note 1: Non-MG-hosted endpoints (legacy trunks) do not require a separate TID allocation step. Provisioning of table TRKMEM with the carrier channel is all that is required.

Note 2: The IID table on the respective GWC must be populated with the corresponding endpoint/channel data for MG-based PRI trunks.

Note 3: The remaining tables on the CS2000 for trunk provisioning (i.e. CLLI, TRKGRP, etc.) are unchanged by this activity and are still managed through Table Editor.

9.1 Supported Trunk Provisioning Command

Following XML commands are supported for Trunk Provisioning use OSSGate.

- AddTuple
- DelTuple
- ReplaceTuple
- GetTuple
- GetRange
- GetTupleWithoutGwcemCheck
- GetRangeWithoutGwcemCheck

9.2 Terminology, Description of Method Parameters

- **TableName** - Name of the table. TYPE *string*. Min length 1, Max length 8. The valid values include "TRKSGRP" and "TRKMEM".
- **Tuple** - Information of the Tuple data. TYPE *string*.

- **MGName** - Name of the Media Gateway. TYPE *string*.
- **EndPointName** - Name of the Endpoint. TYPE *string*.
- **BackupMGName** - Name of the backup Media Gateway. TYPE *string*.
- **BackupEndPointName** - Name of the backup Endpoint. TYPE *string*.
- **TupleKey** - Key of the Tuple. TYPE *string*.
- **NumberOfTuples** - Number of the returnedTuples. TYPE *string*. If **NumberOfTuples** is less than 0, the number of the returned tuples is the default value in System.
- **value** - Value of the Return Code. TYPE *string only contains number*.
- **text** - Message of the operation result. TYPE *string*.
- **severity** - Type of the serverity. TYPE *string*. The valid values include "INFORMATION", "WARNING", "MINOR", "MAJOR" and "CRITICAL".
- **msgTxtParm** - Information of the parameters. TYPE *string*.
- **GWCNumber** - No. of the GWC. TYPE *unsigned short*. Min is 0.
- **NodeNumber** - No. of the Node. TYPE *unsigned short*. Min is 0.
- **TerminalNumber** - No. of the Terminal. TYPE *unsigned short*. Min is 0.
- **BackupTerminalNumber** - No. of the Backup Terminal. TYPE *unsigned short*. Min is 0.

The following parameters will be returned when some errors happen.

- **Number** - No. of the error. TYPE *string only contains number*. Valid values include "14100", "14101", "14102", "14103", "14104", "14105", "14107" and "14108".
- **Message** - Message of the error. TYPE *string*.
- **Severity** - Serverity of the error. TYPE *string*. The valid values include "INFORMATION", "WARNING", "MINOR", "MAJOR" and "CRITICAL".
- **Param*i*** - Parameter, *i* is *integer*. TYPE *string*.
- **name** - Name of the Sub Component. TYPE *string*.

- **value** - Value of the Return Code. TYPE *integer*.
- **text** - Message of the operation result. TYPE *string*.

9.3 Description of Method Parameters

1. **AddTuple** - Interface/Method that adds a tuple to the respective table.
 - Input data:
 - Version - Optional
 - TableName - Required
 - Tuple - Required
 - MGName - Optional
 - EndPointName - Optional
 - BackupMGName - Optional
 - BackupEndPointName - Optional
 - Output data:
 - tableName - Required
 - value - Required. TYPE *string only contains numbers*.
 - text - Optional
 - severity - Optional
 - msgTxtParm - Optional
2. **DelTuple** - Interface/Method that deletes a tuple from the respective table based on the tuple key or the full tuple.
 - Input data:
 - Version - Optional
 - TableName - Required
 - TupleKey - Required
 - Output data:
 - tableName - Required
 - value - Required. TYPE *string only contains numbers*.
 - text - Optional
 - severity - Optional
 - msgTxtParm - Optional

3. ReplaceTuple - Interface/Method that changes a tuple based on the tuple key or the full tuple.

- Input data:

- Version - Optional
- TableName - Required
- Tuple - Required
- MGName - Optional
- EndPointName - Optional
- BackupMGName - Optional
- BackupEndPointName - Optional

- Output data:

- tableName - Required
- value - Required. TYPE *string only contains numbers*.
- text - Optional
- severity - Optional
- msgTxtParm - Optional

4. GetTuple - Interface/Method that lists or queries a specific tuple based on the tuple key or full tuple from the respective table. If no key or tuple is provided, the first tuple in the specified table will be returned to the OSS.

- Input data:

- Version - Optional
- TableName - Required
- TupleKey - Required

- Output data:

- tableName - Required
- value - Required. TYPE *string only contains numbers*.
- text - Optional
- severity - Optional
- msgTxtParm - Optional
- Tuple - Optional
- GWCNumber - Optional
- NodeNumber - Optional

- TerminalNumber - Optional
- BackupTerminalNumber - Optional
- 5. **GetRange** - Interface/Method that lists the specified number of tuples from the respective table based on the tuple key or the full tuple of the first tuple in the range and the number of tuples requested. If a tuple key is not provided, the list of tuples will be captured from the beginning or “top” of the specified table.
 - Input data:
 - Version - Optional
 - TableName - Required
 - TupleKey - Required
 - NumberOfTuples - Optional
 - Output data:
 - tableName - Required
 - value - Required. TYPE *string only contains numbers*.
 - text - Optional
 - severity - Optional
 - msgTxtParm - Optional
 - Tuple - Optional
 - GWCNumber - Optional
 - NodeNumber - Optional
 - TerminalNumber - Optional
 - BackupTerminalNumber - Optional
- 6. **GetTupleWithoutGwcemCheck** - Interface/Method that lists or queries a specific tuple based on the tuple key or full tuple from the respective table without GWCEM check. If no key or tuple is provided, the first tuple in the specified table will be returned to the OSS.
 - Input data:
 - Version - Optional
 - TableName - Required
 - TupleKey - Required
 - Output data:
 - tableName - Required

- value - Required. TYPE *string only contains numbers*.
 - text - Optional
 - severity - Optional
 - msgTxtParm - Optional
 - Tuple - Optional
 - GWCNumber - Optional
 - NodeNumber - Optional
 - TerminalNumber - Optional
 - BackupTerminalNumber - Optional
7. **GetRangeWithoutGwcemCheck** - Interface/Method that lists the specified number of tuples from the respective table based on the tuple key or the full tuple of the first tuple in the range and the number of tuples requested without GWCEM check. If a tuple key is not provided, the list of tuples will be captured from the beginning or “top” of the specified table.
- Input data:
 - Version - Optional
 - TableName - Required
 - TupleKey - Required
 - NumberOfTuples - Required
 - Output data:
 - tableName - Required
 - value - Required. TYPE *string only contains numbers*.
 - text - Optional
 - severity - Optional
 - msgTxtParm - Optional
 - Tuple - Optional
 - GWCNumber - Optional
 - NodeNumber - Optional
 - TerminalNumber - Optional
 - BackupTerminalNumber - Optional

9.4 XML Commands

The following are examples of XML commands and the corresponding response messages for each operation supported for ADSL flow through pro-

visioning. The XML coding in these examples is formatted for ease of understanding.

Figure 147 An example of AddTuple request without MGName and EndpointName:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>TrunkProv</Interface>
    <Methods Version="2.0">
      <AddTuple TableName="TRKMEM" Tuple="SUC101ISUPV2LP 1 0 GWC 5 18 1"/>
    </Methods>
  </Command>
</CommandList>
```

Figure 148 An example of AddTuple response without MGName and EndpointName:

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkProv</Interface>
    <Methods>
      <AddTuple>
        <ReturnCode tableName="TRKMEM" value="0" text="Command Successful!"/>
      </AddTuple>
    </Methods>
  </Response>
</CommandList>
```

Figure 149 An example of AddTuple request with MGName and EndpointName:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>TrunkProv</Interface>
    <Methods Version="2.0">
      <AddTuple TableName="TRKMEM" Tuple="PRIFORUPGRADED 1 0
        GWC PRIFORDOC DS1_10.1"
        MGName="PRIFORDOC"
        EndPointName="DS1_10.1"/>
    </Methods>
  </Command>
</CommandList>
```

Figure 150 An example of AddTuple response for Succession trunk:

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkProv</Interface>
    <Methods>
      <AddTuple>
        <ReturnCode tableName="TRKMEM" value="0" text="Command Successful!"/>
      </AddTuple>
    </Methods>
  </Response>
</CommandList>
```

Figure 151 An example of DelTuple request:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>TrunkProv</Interface>
    <Methods Version="2.0">
      <DelTuple TableName="TRKMEM" TupleKey="IBNT2ISUPOG 0"/>
    </Methods>
  </Command>
</CommandList>
```

Figure 152 An example of DelTuple response:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Response>
    <Interface>TrunkProv</Interface>
    <Methods>
      <DelTuple>
        <ReturnCode TableName="TRKMEM" value="0" text="Command Successful!"/>
      </DelTuple>
    </Methods>
  </Response>
</CommandList>
```

Figure 153 An example of ReplaceTuple request without MGName and EndpointName:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>TrunkProv</Interface>
    <Methods Version="2.0">
      <ReplaceTuple TableName="TRKMEM"
        Tuple="PRIFORUPGRADED0C 3 0 GWC 0 67 38"/>
    </Methods>
  </Command>
</CommandList>
```

Figure 154 An example of ReplaceTuple response without MGName and EndpointName:

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkProv</Interface>
    <Methods>
      <ReplaceTuple>
        <ReturnCode tableName="TRKMEM" value="0" text="Command Successful!"/>
      </ReplaceTuple>
    </Methods>
  </Response>
</CommandList>
```

Figure 155 An example of ReplaceTuple request with MGName and EndpointName:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>TrunkProv</Interface>
    <Methods Version="2.0">
      <ReplaceTuple TableName="TRKMEM" Tuple="PRIFORUPGRADED
        3 0 GWC PRIFORDOC DS1_10.6"
        MGName="PRIFORDOC"
        EndPointName="DS1_10.6"/>
    </Methods>
  </Command>
</CommandList>
```

Figure 156 An example of ReplaceTuple response with MGName and EndpointName:

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkProv</Interface>
    <Methods>
      <ReplaceTuple>
        <ReturnCode tableName="TRKMEM" value="0" text="Command Successful!"/>
      </ReplaceTuple>
    </Methods>
  </Response>
</CommandList>
```

Figure 157 An example of GetTuple request:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>TrunkProv</Interface>
    <Methods Version="2.0">
      <GetTuple TableName="TRKSGRP" TupleKey="IBNT2ISUPOG 0"/>
    </Methods>
  </Command>
</CommandList>
```

Figure 158 An example of GetTuple response:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Response>
    <Interface>TrunkProv</Interface>
    <Methods>
      <GetTuple>
        <ReturnCode TableName="TRKSGRP" value="0" text="Command Successful!"/>
        <TupleData>
          <Tuple>IBNT2ISUPOG 0 DS1SIG C7UP 2W N N UNEQ NONE Q764
                                THRH 100 DMSNODE $NIL CIC</Tuple>
        </TupleData>
      </GetTuple>
    </Methods>
  </Response>
</CommandList>
```

Figure 159 An example of GetRange request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkProv</Interface>
    <Methods Version="2.0">
      <GetRange
        TableName="TRKMEM"
        TupleKey="PRIFORUPGRADED 1"
        NumberOfTuples="5"/>
    </Methods>
  </Command>
</CommandList>
```

Figure 160 An example of GetRange response:

```

<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkProv</Interface>
    <Methods>
      <GetRange>
        <ReturnCode tableName="TRKMEM" value="0" text="Command Successful!"/>
        <TupleData>
          <Tuple>PRIFORUPGRADED 1 0 PRIFORDOC DS1_10.1 </Tuple>
          <GWCNumber>0</GWCNumber>
          <NodeNumber>67</NodeNumber>
          <TerminalNumber>32</TerminalNumber>
        </TupleData>
        <TupleData>
          <Tuple>PRI 2 0 GAOPRI E1_1001.2 </Tuple>
          <GWCNumber>0</GWCNumber>
          <NodeNumber>67</NodeNumber>
          <TerminalNumber>2</TerminalNumber>
        </TupleData>
        <TupleData>
          <Tuple>PRI 3 0 GAOPRI E1_1001.3 </Tuple>
          <GWCNumber>0</GWCNumber>
          <NodeNumber>67</NodeNumber>
          <TerminalNumber>3</TerminalNumber>
        </TupleData>
        <TupleData>
          <Tuple>H323TEST 2 0 GWC 3 72 2 ; TID not found in GWCEM;
                                node#=72;term#=2.</Tuple>
          <GWCNumber>3</GWCNumber>
          <NodeNumber>72</NodeNumber>
          <TerminalNumber>2</TerminalNumber>
        </TupleData>
        <TupleData>
          <Tuple>H323TEST 3 0 GWC 3 72 3; TID not found in GWCEM;
                                node#=72;term#=3.</Tuple>
          <GWCNumber>3</GWCNumber>
          <NodeNumber>72</NodeNumber>
          <TerminalNumber>3</TerminalNumber>
        </TupleData>
      </GetRange>
    </Methods>
  </Response>
</CommandList>

```

Figure 161 An example of GetTupleWithoutGwcemCheck request:

```

<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>TrunkProv</Interface>
    <Methods Version="2.0">
      <GetTupleWithoutGwcemCheck TableName="TRKSGRP"
                                TupleKey="PVG40_2_PRI_USR 0"/>
    </Methods>
  </Command>
</CommandList>

```


Figure 162 An example of GetTupleWithoutGwcemCheck response:

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkProv</Interface>
    <Methods>
      <GetTupleWithoutGwcemCheck>
        <ReturnCode tableName="TRKSGRP" value="0" text="Command Successful!"/>
        <TupleData>
          <Tuple>PVG40_2_PRI_USR 0 DS1SIG ISDN 20 20 96ISOQSIG 2 N STAND USER
            PT_PT USER N INTERNAL N N 255 N DEFAULT GWC 5 18 47 64K HDLC $ $</Tuple>
          <GWCNumber>5</GWCNumber>
          <NodeNumber>18</NodeNumber>
          <TerminalNumber>47</TerminalNumber>
        </TupleData>
      </GetTupleWithoutGwcemCheck>
    </Methods>
  </Response>
</CommandList>
```

Figure 163 An example of GetRangeWithoutGwcemCheck request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<CommandList>
  <Command>
    <Interface>TrunkProv</Interface>
    <Methods Version="2.0">
      <GetRangeWithoutGwcemCheck TableName="TRKMEM"
        TupleKey="PVG40_2_PRI_USR 1"
        NumberOfTuples="5"/>
    </Methods>
  </Command>
</CommandList>
```

Figure 164 An example of **GetRangeWithoutGwcemCheck** response:

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkProv</Interface>
    <Methods>
      <GetRangeWithoutGwcemCheck>
        <ReturnCode tableName="TRKMEM" value="0" text="Command Successful!"/>
        <TupleData>
          <Tuple>PVG40_2_PRI_USR 1 0 GWC 5 18 32 </Tuple>
          <GWCNumber>5</GWCNumber>
          <NodeNumber>18</NodeNumber>
          <TerminalNumber>32</TerminalNumber>
        </TupleData>
        <TupleData>
          <Tuple>PVG40_2_PRI_USR 2 0 GWC 5 18 33 </Tuple>
          <GWCNumber>5</GWCNumber>
          <NodeNumber>18</NodeNumber>
          <TerminalNumber>33</TerminalNumber>
        </TupleData>
        <TupleData>
          <Tuple>PVG40_2_PRI_USR 3 0 GWC 5 18 34 </Tuple>
          <GWCNumber>5</GWCNumber>
          <NodeNumber>18</NodeNumber>
          <TerminalNumber>34</TerminalNumber>
        </TupleData>
        <TupleData>
          <Tuple>PVG40_2_PRI_USR 4 0 GWC 5 18 35 </Tuple>
          <GWCNumber>5</GWCNumber>
          <NodeNumber>18</NodeNumber>
          <TerminalNumber>35</TerminalNumber>
        </TupleData>
        <TupleData>
          <Tuple>PVG40_2_PRI_USR 5 0 GWC 5 18 36</Tuple>
          <GWCNumber>5</GWCNumber>
          <NodeNumber>18</NodeNumber>
          <TerminalNumber>36</TerminalNumber>
        </TupleData>
      </GetRangeWithoutGwcemCheck>
    </Methods>
  </Response>
</CommandList>
```

Figure 165 An example of error response:

```

<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkProv</Interface>
    <Methods>
      <AddTuple>
        <ReturnCode tableName="TRKMEM" value="707" text="ERROR, An error was received from the
Communication Server.">
          <Subcomponent name="Communication Server" value="20" text="RC=3; TxtIdx=20; CMError="The GWC node number has been
67\015TERMINAL ALREADY BOUND TO ANOTHER TRUNK\015"; end; end/'>
        </ReturnCode>
      </AddTuple>
    </Methods>
  </Response>
</CommandList>

```

9.5 User Authorization for Trunk provisioning Operations

The security for the OSSGate interface includes support for permission or 'Authorization' levels for commands. Each operation is associated with one or more user groups. In order to execute a command, a user must belong to at least one of the associated user groups. The user groups associated with each Trunk Provisioning operation is listed in Table-10 below.

Table 13: Trunk Provisioning user groups

Command	User Group				
	trkadm	trkiprov	trksprov	trkmtc	trlro
GetTuple	X	X	X	X	X
GetRange	X	X	X	X	X
AddTuple	X	X			
ReplaceTuple	X	X			
DelTuple	X	X			
GetTupleWithoutGwcemCheck	X	X	X	X	X
GetRangeWithoutGwcemCheck	X	X	X	X	X

9.6 Limitation and Restrictions

- **Abort Operation Command is Unsupported** - the ability to abort an XML request received from OSSGate is not supported. The Trunk Provisioning application writes to only one data repository and does not need to “roll back” data. Either the XML request is processed successfully, or a corresponding error message is returned to OSSGate. If an OSS client would like to terminate or abort a table operation, the client can break the telnet link to OSSGate.
- **Pause Operation Command is Unsupported**- the ability to pause an XML request received from OSSGate will not be supported. The Trunk Provisioning application resides in a unified framework which cannot pause one application while allowing others to continue unabated. If an OSS client wishes to pause a table operation, the client can abort it by breaking the telnet link to OSSGate. When the client reconnects, the operation will have to be restarted.

9.7 Return Codes

The Trunk Provisioning application will respond to each XML request with an XML response. The XML response will contain a return code that will signify whether the request was processed properly or if an error was incurred.

The return code of “0” means the table operation request was processed successfully.

The following return codes for errors may be sent back if an error occurs during the processing of a table operation request.

Table 14: Trunk Provisioning Return Codes

Operation	Return Code	Description	MsgText	Comment
All Trunk Provisioning operations	0	Successful operation	Varies, depending on the operation and specific scenario.	This error code means that there was no error and the request was processed successfully.

Table 14: Trunk Provisioning Return Codes

Operation	Return Code	Description	MsgText	Comment
All Trunk Provisioning operations	701	Unsuccessful operation. Trunk Provisioning is unavailable.	"The Trunk Provisioning application is currently off-line. Please try your request again later."	This error generally means the Trunk Provisioning application is off-line and should be manually restarted using the administrative control tool.
All Trunk Provisioning operations	702	Unsuccessful operation.	"An error has occurred while attempting to execute the Trunk Provisioning command."	Probably due to catching an unexpected exception. An example is the database connection being down.
All Trunk Provisioning operations	703	Unsuccessful operation. Invalid customer input.	"An invalid XML message was received. Please try your request again."	This error generally means the OSS sent in an invalid message. The OSS should verify their XML using the XSD. If it complies, Nortel GPS should be notified.
All Trunk Provisioning operations	704	Unsuccessful operation. GWCEM error.	"An error has occurred within the GWCEM application."	Probably due to catching an unexpected exception.
Add Tuple Change Tuple	705	Unsuccessful operation.	Varies, depending on the operation and specific scenario.	There is a potential data integrity issue within a database.
Add Tuple	706	Unsuccessful operation. Tuple Already Exists.	Varies, depending on the operation and specific scenario.	It is possible someone back-door added the tuple into the CS (bypassing the CS2000 Management Tools server).

Table 14: Trunk Provisioning Return Codes

Operation	Return Code	Description	MsgText	Comment
All Trunk Provisioning operations	707	Unsuccessful operation.	Varies, depending on the operation and specific scenario.	Major problem affecting many areas of the server and needs to be resolved immediately. (possible loss of connection between the provisioning applications and the Communication Server).
Change Tuple Delete Tuple Query Tuple	708	Unsuccessful operation. Tuple Not Found.	Varies, depending on the operation and specific scenario.	It is possible someone erased the tuple from the Communication Server.
All Trunk Provisioning operations	709	Unsuccessful operation. Unknown internal problem.	Varies, depending on the operation and specific scenario.	
GetRange GetRangeWithoutGwcemCheck	710	Unsuccessful operation. NumberOfTuples Invalid.	Varies, depending on the operation and specific scenario.	
All Trunk Provisioning operations	501	Insufficient security privileges to perform this action.	The User is not authorized to perform the operation	User is not a member of correct authorization group.

10: V5.2 Carrier Provisioning with OSSGate

V5 is a standard interface between the access network and the carrier switch for basic telephony, ISDN and semi-permanent leased lines. A V5.2 Interface consists of a 'bundle' of 'n' E1 carriers where $1 \leq n \leq 16$. OSS can provision the V5.2 interface with OSSGate. The XML template defines the data which the OSS must send in order to provision a V5.2 interface. The data arriving from the OSS is parsed to route it to the right application. The V5.2 configuration data which is likely to be passed from the OSS is as follows:

- The interface ID.
- The number of E1 links in the interface.
- The gateways which these links are on.
- The links which contain the signaling.

Functionality supported by OSSGate for V5.2 interface include add, delete and list V5.2 interfaces.

10.1 Supported V5.2 Provisioning Commands

The following V5.2 Provisioning operations are supported in SN06 and later:

- Add Interface
- Delete Interface
- List Interface
- List All Interfaces
- Add V5Prov Template
- Delete V5Prov Template
- List V5Prov Template
- List All V5Prov Template
- Add V5Signalling Template
- Delete V5Signalling Template
- List V5Signalling Template
- List All V5Signalling Template
- Add V5Ring Template
- Delete V5Ring Template
- List V5Ring Template
- List All V5Ring Template

10.2 Terminology, Description of Method Parameters

- **siteGwcLoc** - A four character site identifier, together with frame and unit numbers that together allow the AN to be identified. Format: <Host identifier> <Frame number=0-511> <Unit number=0-9> Example - LG 02 1
- **gwcId** - This is the number of the GWC on which the V5.2 Interface will reside. Range 0~255. Type *integer*.
- **v52InterfaceId** - A unique value for interface ID. Range 0~16777215. Type *integer*.
- **v5ProvRef** - V5 provisioning identifier, identifies V5 provisioning variant used with the V5.2.
- **v5SigTableRef** - V5 signaling identifier, identifies V5 signaling variant used with the V5.2.
- **v5RingTableRef** - V5 ring identifier, identifies V5 ring variant used with the V5.2.
- **maxlinesSelector** - A selector field used for future support of increased MAXLINES of 3072. Currently, REG is only supported for IP_V52. Possible values are "REG", "PRIM", "SIS1", "SIS2". Type *string*.
- **maxlines** - Maximum lines defined on a V5.2 interface. Range 1~3072. Type *integer*.

Each V5.2 Interface can consists of a 'bundle' of 'n' E1 carriers where $1 \leq n \leq 16$ and each bundle require the following parameters data filled:

- **linkId** - Identifiers to each of the E1 carriers (up to 16) that make up the V5.2 interface. Range 1~4094. Type *integer*.
- **epGrp** - E1 carrier on PVG gateway in Aspen format.

Format:

<GW Name>.E1_<([1-9]|1[0-5])(0[1-9]|1[0-9]|2[0-9]|3[0-2])>

Example: PVG1.E1_1111

- **v5provid** - V5 provisioning variant ID. Range 0~127. Type *integer*.
- **bcctimer** - Two Bearer channel control (BCC) timers sets both timers TBCC1 and TBCC4 to values between 500 and 1500 ms. (1=100ms). Range 5~15. Type *integer*.
- **cchnlinflist** - C-channel link information. Defines on what link and channel the protocol process are carried on. Array of 1-16.
 - chnlid - Control Channel Number. Range 0~65535. Type *integer*.
 - lcc - Link and C-channel.
 - lnk - V5.2 Link Number associated with physical GPP P-side link. Range 1~16. Type *integer*.
 - chnl - C-channel Number. Possible values are 16, 15, 31. Type *integer*.
 - cpathlist - Type of control messaging carried on C-Channel. Possible values are "CTRL", "PSTN", "ISDD", "ISDF", "ISDP", "PSET". Type *string array*.
- **prot1** - Protection Link 1. Secondary link protection group 1 switches to if primary C-channel link fails. Range 1~16. Type *integer*.
- **prot2** - Protection link 2. Standby link and C-channel for protection group 2, first entry being link second entry. Array of up to 3 (lnk, chnl) pairs.
 - lnk - link id. Range 1~16. Type *integer*.
 - chnl - channel. Possible values are 16, 15, 31. Type *integer*.
- **alarmthreshold** - Alarm Threshold level of V5.2 link failures before a major alarm is triggered. Range 0~100. Type *integer*.
- **v5sigid** - A unique identifier for V5Sig. Range string of up to 16 alphanumeric. Type *string*.
- **atten** - Attenuation. Possible values are "V5_NONE", "V5_ANALOG", "V5_DIGITAL". Type *string*.
- **apa** - Accelerated port alignment. Possible values are "N", "Y". Type *string*.
- **plf** - Park line feed. Possible values are "N", "Y". Type *string*.
- **ds1flash** - Digit 1 register recall. Possible values are "N", "Y". Type *string*.
- **eoc** - End of call signaling. Possible values are "N", "Y". Type *string*.
- **suppind** - Suppression indicator. Possible values are

"NO_SUPP", "LE_SUPP", "TE_SUPP", "LE_TE_SUPP". Type *string*.

- **plsdur** - Plus duration type. Range 0~31. Type *integer*.
- **mtrpn** - Meter pulse notification. Possible values are "N", "Y". Type *string*.
- **lroa** - Line reversal on answer. Possible values are "N", "Y", "CHKLN". Type *string*.
- **lrosfd** - Line reversal on seizure and forward disconnect. Possible values are "N", "Y". Type *string*.
- **rngtype** - Provisionable ring type. Possible values are "C3C", "C3D", "C6F", "C5I", "C6R". Type *string*.
- **v5ringid** - A unique identifier for V5Ring. Range string of up to 16 alphanumeric. Type *string*.
- **std** - Standard ring. Range 0~31. Type *integer*.
- **r01~r15** - Ring char 1 to ring char 15. Range 0~31. Type *integer*.
- **version** - The version refers to the version of the method or operation - e.g., the AddInterface operation may have version 1.0 and 2.0, where version 1.0 takes parameters X and Y and version 2.0 takes parms X, Y, and Z. TYPE *decimal*.
- **usn** - Unique Sequence Number. This parameter is introduced to facilitate sending multiple XML commands in one file. This parameter value will be used to match the responses with the requests. TYPE *integer*.
- **RC** - Return Code. A numeric value indicating the result of the attempted operation (see Table 15, for range and definitions of each possible value). TYPE *short*.
- **MsgTxt** - Appropriate text that describes the result of the operation. TYPE *string*.

10.3 Description of Method Parameters

1. **addInterface** - Interface/Method that adds a V5.2 Interface and associated carriers. This method has the following parameters:
 - Input Data:
 - usn
 - version
 - siteGwcLoc - The AN node location.
 - gwclId - This is the number of the GWC on which the V5.2 Interface will reside.

- v52InterfaceId - A unique value for interface ID. Range 0~16777215. Type *integer*.
- v5ProvRef - V5 provisioning template identifier.
- v5SigTableRef - V5 signaling template identifier.
- v5RingTableRef - V5 ring template identifier.
- maxlinesSelector - Possible values are "REG", "PRIM", "SIS1", "SIS2".
- maxlines - Maximum lines defined on a V5.2 interface. Range 1~3072. Type *integer*.

The following data will be specified for each of the E1 carriers. At least one pair of E1 carriers has to be data filled.

Each V5.2 Interface can consists of a 'bundle' of 'n' E1 carriers where $1 \leq n \leq 16$ and each bundle require the following parameters data filled:

Required data for each of E1 carrier:

- linkId - Identifiers to each of the E1 carriers (up to 16) that make up the V5.2 interface. Range 1~4094.
- epGrp - E1 carrier on PVG gateway in Aspen format.
- Output Data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - RC - Return Code
 - MsgTxt - Text (string) detailing specific results of the operation in terms of Success or Failure of the operation and possible indication of failure reason.
- 2. **deleteInterface** - Interface/Method that deletes the specified V5.2 interface and associated carriers. This method has the following parameters:
 - Input Data:
 - usn
 - v52InterfaceId
 - Output Data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - RC - Return Code

- MsgTxt - Text (string) detailing specific results of the operation in terms of Success or Failure of the operation and possible indication of failure reason.
- 3. **listInterface** - Interface/Method that returns a V5.2 interface and associated carriers. This method has the following parameters:
 - Input Data:
 - usn
 - version
 - v52InterfaceId
 - Output Data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - siteGwcLoc - The AN node location.
 - gwclId - This is the number of the GWC on which the V5.2 Interface will reside.
 - v52InterfaceId - A unique value for interface ID. Range 0~16777215. Type *integer*.
 - v5ProvRef - V5 provisioning template identifier.
 - v5SigTableRef - V5 signaling template identifier.
 - v5RingTableRef - V5 ring template identifier.
 - maxlinesSelector - Possible values are "REG", "PRIM", "SIS1", "SIS2".
 - maxlines - Maximum lines defined on a V5.2 interface. Range 1~3072. Type *integer*.
 - RC - Return Code
 - MsgTxt - Text (string) detailing specific results of the operation in terms of Success or Failure of the operation and possible indication of failure reason.

The following data will be returned for each of the E1 carrier:

- linkId - Identifiers to each of the E1 carriers (up to 16) that make up the V5.2 interface. Range 1~4094.
- epGrp - E1 carrier on PVG gateway in Aspen format.
- 4. **listAllInterfaces** - Interface/Method that returns all V5.2 Interfaces and associated carriers. This method has the following parameters:

- Input Data:
 - usn
 - version
- Output Data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - RC - Return Code
 - MsgTxt - Text (string) detailing specific results of the operation in terms of Success or Failure of the operation and possible indication of failure reason.
 - keylist - This is the number of the GWC on which the V5.2 Interface will reside.
- 5. **addV5ProvTemplate** - Interface/Method that adds a V5PROV template. This method has the following parameters:
 - Input Data:
 - usn
 - version
 - v5provid - V5 provisioning variant ID. Range 0~127. Type *integer*.
 - bcctimer - Two Bearer channel control (BCC) timers sets both timers TBCC1 and TBCC4 to values between 500 and 1500 ms. (1=100ms). Range 5~15. Type *integer*.

The following data will be specified for each of the C-channel link information.

- i. chnlid
- ii. Link and C-channel
 - lnk - link id. Range 1~16. Type *integer*.
 - chnl - channel. Possible values are 16, 15, 31. Type *integer*.
- iii. cpathlist - Type of control messaging carried on C-Channel. Possible values are "CTRL", "PSTN", "ISDD", "ISDF", "ISDP", "PSET". Type *string array*.
 - prot1 - Protection Link 1. Range 1~16. Type *integer*.

The following data will be specified for each protection link 2. Prot2 is an array of up to 3 (lnk, chnl) pairs.

- 1. lnk - link id. Range 1~16. Type *integer*.
- 2. chnl - channel. Possible values are 16, 15, 31. Type *integer*.
- alarmthreshold - Alarm Threshold level of V5.2 link failures before a major alarm is triggered. Range 0~100. Type *integer*.
- Output Data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - RC - Return Code
 - MsgTxt - Text (string) detailing specific results of the operation in terms of Success or Failure of the operation and possible indication of failure reason.
- 6. **deleteV5ProvTemplate** - Interface/Method that deletes a V5PROV template. This method has the following parameters:
 - Input Data:
 - usn
 - version
 - v5provid - The V5 provisioning variant ID. Range 0~127.
 - Output Data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - RC - Return Code
 - MsgTxt - Text (string) detailing specific results of the operation in terms of Success or Failure of the operation and possible indication of failure reason.
- 7. **listV5ProvTemplate** - Interface/Method that returns a V5PROV template. This method has the following parameters:
 - Input Data:
 - usn
 - version
 - v5provid - V5 provisioning variant ID. Range 0~127.
 - Output Data:
 - usn (value should be the same as the input)

- version (value should be the same as the input)
- RC - Return Code
- MsgTxt - Text (string) detailing specific results of the operation in terms of Success or Failure of the operation and possible indication of failure reason.
- v5provid - V5 provisioning variant ID. Range 0~127.
- bcctimer - Two Bearer channel control (BCC) timers sets both timers TBCC1 and TBCC4 to values between 500 and 1500 ms. (1=100ms). Range 5~15.

The following data will be specified for each of the C-channel link information.

- 1. chnlid
- 2. Link and C-channel
 - 2.1 lnk - link id. Range 1~16. Type *integer*.
 - 2.2 chnl - channel. Possible values are 16, 15, 31. Type *integer*.
- 3. cpathlist - Type of control messaging carried on C-Channel. Possible values are "CTRL", "PSTN", "ISDD", "ISDF", "ISDP", "PSET". Type *string array*.
- prot1 - Protection Link 1. Range 1~16. Type *integer*.

The following data will be specified for each protection link 2. Prot2 is an array of up to 3 (lnk, chnl) pairs.

- 1. lnk - link id. Range 1~16. Type *integer*.
- 2. chnl - channel. Possible values are 16, 15, 31. Type *integer*.
- alarmthreshold - Alarm Threshold level of V5.2 link failures before a major alarm is triggered. Range 0~100. Type *integer*.

8. listAllV5ProvTemplate - Interface/Method that returns all V5PROV templates. This method has the following parameters:

- Input Data:

- usn
- version

- Output Data:

- usn (value should be the same as the input)
- version (value should be the same as the input)

- keylist - V5 provisioning variant ID. Range 0~127.
- 9. **addV5SigTemplate** - Interface/Method that adds a V5SIG template. This method has the following parameters:
 - Input Data:
 - usn
 - version
 - v5sigid - A unique identifier for V5Sig. Range string of up to 16 alphanumeric. Type *string*.
 - atten- Attenuation. Possible values are "V5_NONE", "V5_ANALOG", "V5_DIGITAL". Type *string*
 - apa - Accelerated port alignment. Possible values are "N", "Y". Type *string*.
 - plf - Park line feed. Possible values are "N", "Y". Type *string*.
 - ds1flash - Digit 1 register recall. Possible values are "N", "Y". Type *string*.
 - eoc - End of call signaling. Possible values are "N", "Y". Type *string*.
 - suppind - Suppression indicator. Possible values are "NO_SUPP", "LE_SUPP", "TE_SUPP", "LE_TE_SUPP". Type *string*.
 - plsdur - Plus duration type. Range 0~31. Type *integer*.
 - mtrpn - Meter pulse notification. Possible values are "N", "Y". Type *string*.
 - lroa- Line reversal on answer. Possible values are "N", "Y", "CHKLN". Type *string*.
 - lrosfd - Line reversal on seizure and forward disconnect. Possible values are "N", "Y". Type *string*.
 - rngtype - Provisionable ring type. Possible values are "C3C", "C3D", "C6F", "C5I", "C6R". Type *string*.
 - Output Data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - RC - Return Code
 - MsgTxt - Text (string) detailing specific results of the operation in terms of Success or Failure of the operation and possible indication of failure reason.

10. deleteV5SigTemplate - Interface/Method that deletes a V5SIG template. This method has the following parameters:

- Input Data:

- usn
- version
- v5sigid - The unique identifier for V5Sig.

- Output Data:

- usn (value should be the same as the input)
- version (value should be the same as the input)
- RC - Return Code
- MsgTxt - Text (string) detailing specific results of the operation in terms of Success or Failure of the operation and possible indication of failure reason.

11. listV5SigTemplate - Interface/Method that returns a V5SIG template. This method has the following parameters:

- Input Data:

- usn
- version
- v5sigid - The unique identifier for V5Sig.

- Output Data:

- usn (value should be the same as the input)
- version (value should be the same as the input)
- RC - Return Code
- MsgTxt - Text (string) detailing specific results of the operation in terms of Success or Failure of the operation and possible indication of failure reason.
- v5sigid - A unique identifier for V5Sig.
- atten - Attenuation. Possible values are "V5_NONE", "V5_ANALOG", "V5_DIGITAL".
- apa - Accelerated port alignment. Possible values are "N", "Y".
- plf - Park line feed. Possible values are "N", "Y".
- ds1flash - Digit 1 register recall. Possible values are "N", "Y".
- eoc - End of call signaling. Possible values are "N", "Y".

- `suppind` - Suppression indicator. Possible values are "NO_SUPP", "LE_SUPP", "TE_SUPP", "LE_TE_SUPP".
- `plsdur` - Plus duration type. Range 0~31.
- `mtrpn` - Meter pulse notification. Possible values are "N", "Y".
- `lroa` - Line reversal on answer. Possible values are "N", "Y", "CHKLN".
- `lrosfd` - Line reversal on seizure and forward disconnect. Possible values are "N", "Y".
- `rngtype` - Provisionable ring type. Possible values are "C3C", "C3D", "C6F", "C5I", "C6R".
- `ssonhook` - Signal SS: On-hook message flag. Possible values are "N", "Y".

12. `listAllV5SigTemplates` - Interface/Method that returns all V5SIG templates. This method has the following parameters:

- Input Data:

- `usn`
- `version`

- Output Data:

- `usn` (value should be the same as the input)
- `version` (value should be the same as the input)
- `RC` - Return Code
- `MsgTxt` - Text (string) detailing specific results of the operation in terms of Success or Failure of the operation and possible indication of failure reason.
- `keylist` - The unique identifier for V5Sig.

13. `addV5RingTemplate` - Interface/Method that adds a V5RING template. This method has the following parameters

- Input Data:

- `usn`
- `version`
- `v5ringid` - A unique identifier for V5Ring. Range string of up to 16 alphanumeric. Type *string*.
- `std` - Standard ring. Range 0~31. Type *integer*.
- `r01 ~ r15` - Ring char 1 to ring char 15. Range 0~31. Type *integer*.

- Output Data:

- usn (value should be the same as the input)
- version (value should be the same as the input)
- RC - Return Code
- MsgTxt - Text (string) detailing specific results of the operation in terms of Success or Failure of the operation and possible indication of failure reason.

14. deleteV5RingTemplate - Interface/Method that deletes a V5RING template. This method has the following parameters:

- Input Data:

- usn
- version
- v5ringid - The unique identifier for V5Ring.

- Output Data:

- usn (value should be the same as the input)
- version (value should be the same as the input)
- RC - Return Code
- MsgTxt - Text (string) detailing specific results of the operation in terms of Success or Failure of the operation and possible indication of failure reason.

15. listV5RingTemplate - Interface/Method that gets the configuration data for a V5.2 Ring Template.

- Input Data:

- usn
- version
- v5ringid - The unique identifier for V5Ring.

- Output Data:

- usn (value should be the same as the input)
- version (value should be the same as the input)
- RC - Return Code
- MsgTxt - Text (string) detailing specific results of the operation in terms of Success or Failure of the operation and possible indication of failure reason.
- v5ringid - The unique identifier for V5Ring.
- std - Standard ring. Range 0~31.

- r01 ~ r15 - Ring char 1 to ring char 15. Range 0~31.

16. listAllV5RingTemplates - Interface/Method that returns all V5RING templates. This method has the following parameters:

- Input Data:
 - usn
 - version
- Output Data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - RC - Return Code
 - MsgTxt - Text (string) detailing specific results of the operation in terms of Success or Failure of the operation and possible indication of failure reason.
 - keylist - The unique identifier for V5Ring.

10.4 XML Commands

The following are examples of XML commands and the corresponding response messages for each operation supported for V5.2. The XML coding in these examples is formatted for ease of understanding.

Operation addInterface

Figure 166 An example of addInterface request:

```

<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Command>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <addInterface usn="1" version="1.0">
        <Parameters>
          <siteGwcLoc>LG 02 1</siteGwcLoc>
          <gwcd>73</gwcd>
          <v52InterfaceId>123</v52InterfaceId>
          <linkMapTable>
            <linkId>1</linkId>
            <epGrp>PVG1.E1_705</epGrp>
            <linkId>2</linkId>
            <epGrp>PVG1.E1_706</epGrp>
          </linkMapTable>
          <v5ProvRef>20</v5ProvRef>
          <v5SigTableRef>DEFAULT</v5SigTableRef>
          <v5RingTableRef>DEFAULT</v5RingTableRef>
          <maxlinesSelector>REG</maxlinesSelector>
          <maxlines>100</maxlines>
        </Parameters>
      </addInterface>
    </Methods>
  </Command>
</CommandList>

```

Figure 167 An example of **addInterface** response:

```

<?xml version="1.0" encoding="UTF-8"?>
<CommandList >
  <Response>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <addInterface usn="1" version="1.0">
        <RC>0</RC>
        <MsgTxt>Table operation was successful</MsgTxt>
        <ReturnData>
          <addInterface>
        </addInterface>
      </Methods>
    </Response>
  </CommandList>

```

Operation deleteInterface

Figure 168 An example of **deleteInterface** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <deleteInterface usn="1" version="1.0">
        <Parameters>
          <v52InterfaceId>123</v52InterfaceId>
        </Parameters>
      </deleteInterface>
    </Methods>
  </Command>
</CommandList>
```

Figure 169 An example format of **deleteInterface** response:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList >
  <Response>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <deleteInterface usn="1" version="1.0">
        <RC>0</RC>
        <MsgTxt>Table operation was successful</MsgTxt>
        <ReturnData>
        </deleteInterface>
      </Methods>
    </Response>
  </CommandList>
```

Operation listInterface

Figure 170 An example of **listInterface** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <listInterface usn="1" version="1.0">
        <Parameters>
          <v52InterfaceId>123</v52InterfaceId>
        </Parameters>
      </listInterface>
    </Methods>
  </Command>
</CommandList>
```

Figure 171 An example of **listInterface** response:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList >
  <Response>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <listInterface usn="1" version="1.0">
        <RC>0</RC>
        <MsgTxt>Table operation was successful</MsgTxt>
        <ReturnData>
          <siteGwcLoc>LG 02 1</siteGwcLoc>
          <gwcd>73</gwcd>
          <v52InterfaceId>123</v52InterfaceId>
          <linkMapTable>
            <linkId>1</linkId>
            <epGrp>PVG1.E1_705</epGrp>
            <linkId>2</linkId>
            <epGrp>PVG1.E1_706</epGrp>
          </linkMapTable>
          <v5ProvRef>20</v5ProvRef>
          <v5SigTableRef>DEFAULT</v5SigTableRef>
          <v5RingTableRef>DEFAULT</v5RingTableRef>
          <maxlinesSelector>REG</maxlinesSelector>
          <maxlines>100</maxlines>
        </ReturnData>
      </listInterface>
    </Methods>
  </Response>
</CommandList>
```

Operation listAllInterfacesFigure 172 An example of **listAllInterfaces** request:

```

<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <listAllInterfaces usn="1" version="1.0">
        <Parameters>
          </Parameters>
        </listAllInterfaces>
      </Methods>
    </Command>
  </CommandList>

```

Figure 173 An example of **listAllInterfaces** response:

```

<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Response>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <listAllInterfaces usn="1" version="1.0">
        <RC>0</RC>
        <MsgTxt>Table operation was successful</MsgTxt>
        <ReturnData>
          <keylist>73</keylist>
          <keylist>74</keylist>
        </ReturnData>
      </listAllInterfaces>
    </Methods>
  </Response>
</CommandList>

```


Operation addV5ProvTemplate

Figure 174 An example of addV5ProvTemplate request:

```

<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <addV5ProvTemplate usn="1" version="1.0">
        <Parameters>
          <v5provid>66</v5provid>
          <bcctimer>5</bcctimer>
          <cchnlinflist>
            <chnlid>0</chnlid>
            <lcc>
              <lnk>1</lnk>
              <chnl>16</chnl>
            </lcc>
            <cpathlist>CTRL</cpathlist>
            <chnlid>1</chnlid>
            <lcc>
              <lnk>3</lnk>
              <chnl>16</chnl>
            </lcc>
            <cpathlist>PSTN</cpathlist>
          </cchnlinflist>
          <prot1>2</prot1>
          <prot2>
            <lnk>4</lnk>
            <chnl>16</chnl>
          </prot2>
          <alarmthreshold>20</alarmthreshold>
        </Parameters>
      </addV5ProvTemplate>
    </Methods>
  </Command>
</CommandList>

```

Figure 175 An example of addV5ProvTemplate response:

```

<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Response>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <addV5ProvTemplate usn="1" version="1.0">
        <RC>0</RC>
        <MsgTxt>Table operation was successful</MsgTxt>
        <ReturnData>
        </addV5ProvTemplate>
      </Methods>
    </Response>
  </CommandList>

```

Operation deleteV5ProvTemplate

Figure 176 An example of **deleteV5ProvTemplate** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <deleteV5ProvTemplate usn="1" version="1.0">
        <Parameters>
          <v5ProvId>66</v5ProvId>
        </Parameters>
      </deleteV5ProvTemplate>
    </Methods>
  </Command>
</CommandList>
```

Figure 177 An example of **deleteV5ProvTemplate** response:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Response>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <deleteV5ProvTemplate usn="1" version="1.0">
        <RC>0</RC>
        <MsgTxt>Table operation was successful</MsgTxt>
        <ReturnData>
        </ReturnData>
      </deleteV5ProvTemplate>
    </Methods>
  </Response>
</CommandList>
```

Operation listV5ProvTemplate

Figure 178 An example of **listV5ProvTemplate** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <listV5ProvTemplate usn="1" version="1.0">
        <Parameters>
          <v5ProvId>66</v5ProvId>
        </Parameters>
      </listV5ProvTemplate>
    </Methods>
  </Command>
</CommandList>
```

Figure 179 An example of **listV5ProvTemplate** response:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList >
  <Response>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <listV5ProvTemplate usn="1" version="1.0">
        <RC>0</RC>
        <MsgTxt>Table operation was successful</MsgTxt>
        <ReturnData>
          <v5provid>66</v5provid>
          <bcctimer>5</bcctimer>
          <cchnlinflist>
            <chnlid>0</chnlid>
            <lcc>
              <lnk>1</lnk>
              <chnl>16</chnl>
            </lcc>
            <cpathlist>
              <cpathlist>CTRL</cpathlist>
            </cpathlist>
            <chnlid>1</chnlid>
            <lcc>
              <lnk>3</lnk>
              <chnl>16</chnl>
            </lcc>
            <cpathlist>
              <cpathlist>PSTN</cpathlist>
            </cpathlist>
          </cchnlinflist>
          <prot1>2</prot1>
          <prot2>
            <lnk>4</lnk>
            <chnl>16</chnl>
          </prot2>
          <alarmthreshold>20</alarmthreshold>
        </ReturnData>
      </listV5ProvTemplate>
    </Methods>
  </Response>
</CommandList>
```

Operation listAllV5ProvTemplatesFigure 180 An example of **listAllV5ProvTemplates** request:

```

<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <listAllV5ProvTemplates usn="1" version="1.0">
        <Parameters>
          </Parameters>
        </listAllV5ProvTemplates>
      </Methods>
    </Command>
  </CommandList>

```

Figure 181 An example of **listAllV5ProvTemplates** response:

```

<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Response>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <listAllV5ProvTemplates usn="1" version="1.0">
        <RC>0</RC>
        <MsgTxt>Table operation was successful</MsgTxt>
        <ReturnData>
          <keylist>1</keylist>
          <keylist>4</keylist>
          <keylist>66</keylist>
        </ReturnData>
      </listAllV5ProvTemplates>
    </Methods>
  </Response>
</CommandList>

```

Operation addV5SigTemplate

Figure 182 An example of addV5SigTemplate request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <addV5SigTemplate usn="1" version="1.0">
        <Parameters>
          <v5sigid>SIG1</v5sigid>
          <atten>V5_ANALOG</atten>
          <apa>N</apa>
          <plf>N</plf>
          <ds1flash>N</ds1flash>
          <eoc>N</eoc>
          <suppind>NO_SUPP</suppind>
          <plsdur>5</plsdur>
          <mtrpn>N</mtrpn>
          <lroa>N</lroa>
          <lrosfd>N</lrosfd>
          <rngtype>C3C</rngtype>
        </Parameters>
      </addV5SigTemplate>
    </Methods>
  </Command>
</CommandList>
```

Figure 183 An example of addV5SigTemplate response:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Response>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <addV5SigTemplate usn="1" version="1.0">
        <RC>0</RC>
        <MsgTxt>Table operation was successful</MsgTxt>
        <ReturnData>
        </addV5SigTemplate>
      </Methods>
    </Response>
  </CommandList>
```

Operation deleteV5SigTemplate

Figure 184 An example of **deleteV5SigTemplate** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <deleteV5SigTemplate usn="1" version="1.0">
        <Parameters>
          <v5sigid>SIG1</v5sigid>
        </Parameters>
      </deleteV5SigTemplate>
    </Methods>
  </Command>
</CommandList>
```

Figure 185 An example of **deleteV5SigTemplate** response:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Response>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <deleteV5SigTemplate usn="1" version="1.0">
        <RC>0</RC>
        <MsgTxt>Table operation was successful</MsgTxt>
        <ReturnData>
        </ReturnData>
      </deleteV5SigTemplate>
    </Methods>
  </Response>
</CommandList>
```

Operation listV5SigTemplateFigure 186 An example of **listV5SigTemplate** request:

```

<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <listV5SigTemplate usn="1" version="1.0">
        <Parameters>
          <v5sigid>SIG1</v5sigid>
        </Parameters>
      </listV5SigTemplate>
    </Methods>
  </Command>
</CommandList>

```

Figure 187 An example format of **listV5SigTemplate** response:

```

<?xml version="1.0" encoding="UTF-8"?>
<CommandList >
  <Response>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <listV5SigTemplate usn="1" version="1.0">
        <RC>0</RC>
        <MsgTxt>Table operation was successful</MsgTxt>
        <ReturnData>
          <v5sigid>SIG1</v5sigid>
          <atten>V5_ANALOG</atten>
          <apa>N</apa>
          <plf>N</plf>
          <ds1flash>N</ds1flash>
          <eoc>N</eoc>
          <suppind>NO_SUPP</suppind>
          <plsdur>5</plsdur>
          <mtrpn>N</mtrpn>
          <lroa>N</lroa>
          <lrosfd>N</lrosfd>
          <rngtype>C3C</rngtype>
          <ssonhook>N</ssonhook>
        </ReturnData>
      </listV5SigTemplate>
    </Methods>
  </Response>
</CommandList>

```

Operation listAllV5SigTemplatesFigure 188 An example of **listAllV5SigTemplates** request:

```

<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <listAllV5SigTemplates usn="1" version="1.0">
        <Parameters>
          </Parameters>
        </listAllV5SigTemplates>
      </Methods>
    </Command>
  </CommandList>

```

Figure 189 An example of **listAllV5SigTemplates** response:

```

<?xml version="1.0" encoding="UTF-8"?>
<CommandList >
  <Response>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <listAllV5SigTemplates usn="1" version="1.0">
        <RC>0</RC>
        <MsgTxt>Table operation was successful</MsgTxt>
        <ReturnData>
          <keylist>DEFAULT</keylist>
          <keylist>1</keylist>
          <keylist>2</keylist>
          <keylist>ETS</keylist>
          <keylist>SIG1</keylist>
          <keylist>SIG2</keylist>
          <keylist>SIG3</keylist>
          <keylist>SIG4</keylist>
        </ReturnData>
      </listAllV5SigTemplates>
    </Methods>
  </Response>
</CommandList>

```


Operation addV5RingTemplate

Figure 190 An example of addV5RingTemplate request:

```

<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <addV5RingTemplate usn="1" version="1.0">
        <Parameters>
          <v5ringid>3</v5ringid>
          <std>3</std>
          <r01>1</r01>
          <r02>1</r02>
          <r03>1</r03>
          <r04>1</r04>
          <r05>1</r05>
          <r06>1</r06>
          <r07>1</r07>
          <r08>1</r08>
          <r09>1</r09>
          <r10>1</r10>
          <r11>1</r11>
          <r12>1</r12>
          <r13>1</r13>
          <r14>1</r14>
          <r15>1</r15>
        </Parameters>
      </addV5RingTemplate>
    </Methods>
  </Command>
</CommandList>

```

Figure 191 An example of addV5RingTemplate response:

```

<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Response>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <addV5RingTemplate usn="1" version="1.0">
        <RC>0</RC>
        <MsgTxt>Table operation was successful</MsgTxt>
        <ReturnData>
        </addV5RingTemplate>
      </Methods>
    </Response>
  </CommandList>

```

Operation deleteV5RingTemplate

Figure 192 An example of **deleteV5RingTemplate** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <deleteV5RingTemplate usn="1" version="1.0">
        <Parameters>
          <v5ringid>3</v5ringid>
        </Parameters>
      </deleteV5RingTemplate>
    </Methods>
  </Command>
</CommandList>
```

Figure 193 An example of **deleteV5RingTemplate** response:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Response>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <deleteV5RingTemplate usn="1" version="1.0">
        <RC>0</RC>
        <MsgTxt>Table operation was successful</MsgTxt>
        <ReturnData>
        </ReturnData>
      </deleteV5RingTemplate>
    </Methods>
  </Response>
</CommandList>
```

Operation listV5RingTemplate

Figure 194 An example of **listV5RingTemplate** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <listV5RingTemplate usn="1" version="1.0">
        <Parameters>
          <v5ringid>3</v5ringid>
        </Parameters>
      </listV5RingTemplate>
    </Methods>
  </Command>
</CommandList>
```

Figure 195 An example of **listV5RingTemplate** response:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommandList >
  <Response>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <listV5RingTemplate usn="1" version="1.0">
        <RC>0</RC>
        <MsgTxt>Table operation was successful</MsgTxt>
        <ReturnData>
          <v5ringid>3</v5ringid>
          <std>3</std>
          <r01>1</r01>
          <r02>1</r02>
          <r03>1</r03>
          <r04>1</r04>
          <r05>1</r05>
          <r06>1</r06>
          <r07>1</r07>
          <r08>1</r08>
          <r09>1</r09>
          <r10>1</r10>
          <r11>1</r11>
          <r12>1</r12>
          <r13>1</r13>
          <r14>1</r14>
          <r15>1</r15>
        </ReturnData>
      </listV5RingTemplate>
    </Methods>
  </Response>
</CommandList>
```

Operation listAllV5RingTemplatesFigure 196 An example of **listAllV5RingTemplates** request:

```

<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <listAllV5RingTemplates usn="1" version="1.0">
        <Parameters>
          </Parameters>
        </listAllV5RingTemplates>
      </Methods>
    </Command>
  </CommandList>

```

Figure 197 An example of **listAllV5RingTemplates** response:

```

<?xml version="1.0" encoding="UTF-8"?>
<CommandList >
  <Response>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <listAllV5RingTemplates>
        <RC>0</RC>
        <MsgTxt>Table operation was successful</MsgTxt>
        <ReturnData>
          <keylist>DEFAULT</keylist>
          <keylist>3</keylist>
          <keylist>TEST1</keylist>
          <keylist>TEST</keylist>
        </ReturnData>
      </listAllV5RingTemplates>
    </Methods>
  </Response>
</CommandList>

```

10.4.1 Error handling

V5.2 operations can send error message back to client when an operation fails. The error response has a return code with a message text describing the reason.

Figure 198 Example of an error scenario for list All Interfaces

```

<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Response>
    <Interface>v5CfgMgrIf</Interface>
    <Methods>
      <listAllInterfaces usn="1" version="1.0">
        <RC>8</RC>
        <MsgTxt>V5CFGMGR_INVALID_INPUT</MsgTxt>
        <ReturnData>
        </ReturnData>
      </listAllInterfaces>
    </Methods>
  </Response>
</CommandList>

```

10.5 Authorization for V5.2 Operations

In addition to users belonging to “succssn” group to login to OSSGate, user need to be in application specific groups to perform specific operations. Each operation is associated with one or more user groups. In order to execute a command, a user must belong to at least one of the associated user groups. The user groups associated with each OSSGate V5.2 provisioning operation is listed in Table-12 below.

Table 15: V5.2 User groups

Commands	User Groups									
	trkadm	trkrw	trkmtc	trksprov	trkro	lnadm	lnrw	lnmtc	lnsprov	lnro
addInterface	X	X				X	X			
deleteInterface	X	X				X	X			
listAllInterfaces	X	X	X	X	X	X	X	X	X	X
listInterface	X	X	X	X	X	X	X	X	X	X
addV5ProvTemplate	X	X				X	X			
deleteV5ProvTemplate	X	X				X	X			
listV5ProvTemplate	X	X	X	X	X	X	X	X	X	X
listAllV5ProvTemplates	X	X	X	X	X	X	X	X	X	X
addV5SigTemplate	X	X				X	X			

Table 15: V5.2 User groups

Commands	User Groups									
	trkadm	trkrw	trkmtc	trksprov	trkro	lnadm	lnrw	lnmtc	lnsprov	lnro
deleteV5SigTemplate	X	X				X	X			
listV5SigTemplate	X	X	X	X	X	X	X	X	X	X
listAllV5SigTemplates	X	X	X	X	X	X	X	X	X	X
addV5RingTemplate	X	X				X	X			
deleteV5RingTemplate	X	X				X	X			
listV5RingTemplate	X	X	X	X	X	X	X	X	X	X
listAllV5RingTemplates	X	X	X	X	X	X	X	X	X	X

10.6 V5.2 Return Codes

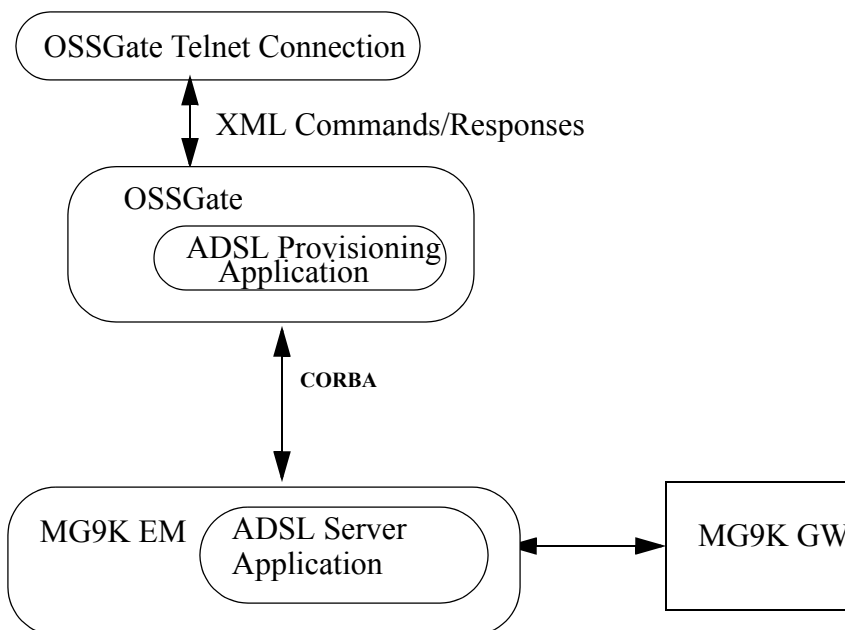
Table 16: V5.2 operational return codes

Return Code	Meaning
0	Successful operation
1	More data coming
2	Request Rejected due to Resource Limits Reached
3	Application was Commanded to Abort
4	Invalid input from client
5	Failed SERVINV update or could not get node name and number
6	Failed Add to GWCEM
7	Failed to update NV
8	Failed Rollback of ADD GWC
9	Query GWC Operation Failed to Get Data from GWC EM
10	Query GWC Operation Failed to Read GWC List from the Network View

11: ADSL Flow through Provisioning for MG9000

The ADSL flow through provisioning application enables authenticated OSS clients to perform provisioning operations on ADSL data circuits on MG9000 through MG9000 EM from CS2000 Management tool server. The provisioning operations include querying, adding, modifying and deleting ADSL subscriber data. Access to this functionality is by connection to OSSGate.

Figure 199 ADSL Provisioning in the Succession Network



11.1 Supported ADSL Provisioning Commands

Following XML commands are supported for ADSL provisioning using OSSGate.

- GetSubscriber
- AddSubscriber
- AddCrossConnection
- ModifySubscriber
- ModifyCrossConnection

- DeleteSubscriber
- DeleteCrossConnection

11.2 Terminology, Description of Method Parameters

- **VMGName** - Name of the media gateway. TYPE *string*
Format: <SITEname 4 character><officeframe=000-511> -
<logicalframe=0-7>-<shelf=0-3>.
Example: LAKE017-6-2.
- **TerminationPoint** - Name of the termination point. This parameter identifies the ADSL data circuit that is being provisioned. TYPE *string*
Format tp/<slot=02-21, 0-padded>/<circuit=0-31, 0-padded>.
Example tp/02/03.
- **version** - The version refers to the version of the method or operation - e.g., the AddSubscriber operation may have version 1.0 and 2.0, where version 1.0 takes parameters X and Y and version 2.0 takes parms X, Y, and Z. TYPE *decimal*.
- **usn** - Unique Sequence Number. This parameter is introduced to facilitate sending multiple XML commands in one file. This parameter value will be used to match the responses with the requests. TYPE *integer*.
- **RC** - Return Code. A numeric value indicating the result of the attempted operation (see Table 19, for range and definitions of each possible value). TYPE *short*.
- **MsgTxt** - Appropriate text that describes the result of the operation. TYPE *string*.
- **DownMaxSpeed** - The value in kbits, Max DownStream 32 - 13376, multiple of 32. TYPE *positiveInteger*.
- **UpMaxSpeed** - The value in kbits, Max Upstream 32 - 1440, multiple of 32. TYPE *unsignedShort*.
- **DownMaxInterleaveDelay** - The value in milliseconds - DownStream delay 10 - 255. TYPE *unsignedShort*.
- **UpMaxInterleaveDelay** - The value in milliseconds - Upstream delay 10 - 255. TYPE *unsignedShort*.
- **DownSignalNoiseMargin** - The value in dB - DownStream noise margin 6 - 31. TYPE *unsignedShort*.
- **UpSignalNoiseMargin** - The value in dB - Upstream noise margin 6 - 31. TYPE *unsignedShort*.

- **TransmissionMode** - Possible values are “Auto Mode”, “ANSI”, “G.DMT”. Default value is “Auto Mode”. TYPE *string*.

Each ADSL data circuit can provide up to 8 cross connections and each cross connection require the following parameters data filled:

- **VPI** - Virtual Path Identifier. Range 16 - 31. TYPE *unsignedShort*.
- **VCI** - Virtual Circuit Identifier. Range 33 - 2047. TYPE *unsignedShort*.
- **UpStreamTrafficDescriptor** - Name of the traffic descriptor user label for transmission. TYPE *string*.
- **DownStreamTrafficDescriptor** - Name of the traffic descriptor user label for reception. TYPE *string*.
- **State** - State of the cross connection. Possible values are “Inactive”, “Active”. TYPE *string*.

11.3 Description of Method Parameters

1. **GetSubscriber** - Interface/Method that gets the configuration data for a subscriber that includes the cross connection data associated with the specified MG and the termination point.
 - Input data:
 - usn
 - version
 - VMGName
 - TerminationPoint
 - Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - DownMaxSpeed - The value in kbits, Max DownStream 32 - 13376, multiple of 32
 - UpMaxSpeed - The value in kbits, Max Upstream 32 - 1440, multiple of 32.
 - DownMaxInterleaveDelay - The value in milliseconds - DownStream delay 10 - 255
 - UpMaxInterleaveDelay - The value in milliseconds - Upstream delay 10 - 255

- DownSignalNoiseMargin - The value in dB - DownStream noise margin 6 - 31
- UpSignalNoiseMargin - The value in dB - Upstream noise margin 6 - 31
- TransmissionMode - Possible values are “Auto Mode”, “ANSI”, “G.DMT”.

The following data will be returned for each of the 8 cross connections.

- i. VCC.ID - Virtual Cross Connection Identifier. The value is range is 1-8 both inclusive.
 - ii. VPI - Virtual Path Identifier. Range 16 - 31.
 - iii. VCI - Virtual Circuit Identifier. Range 33 - 2047.
 - iv. UpStreamTrafficDescriptor - Name of the traffic descriptor user label for transmission.
 - v. DownStreamTrafficDescriptor - Name of the traffic descriptor user label for reception.
 - vi. State - State of the cross connection. Possible values are “Inactive”, “Active”, or “De-provision”.
- RC - Return Code
 - MsgTxt - Text (string) detailing specific results of the operation in terms of Success or Failure of the operation and possible indication of failure reason.
- 2. AddSubscriber** - Interface/Method that will add a new subscriber with the given configuration data. Using this method, multiple cross connections can be configured in one transaction. This method has the following parameters:
- Input data:
 - usn
 - version
 - VMGName
 - TerminationPoint
 - a. Config data (applies to all cross connections)
 - DownMaxSpeed - The value in kbits, Max DownStream 32 - 13376, multiple of 32
 - UpMaxSpeed - The value in kbits, Max Upstream 32 - 1440, multiple of 32.

- DownMaxInterleaveDelay - The value in milliseconds - DownStream delay 10 - 255
- UpMaxInterleaveDelay - The value in milliseconds - Upstream delay 10 - 255
- DownSignalNoiseMargin - The value in dB - DownStream noise margin 6 - 31
- UpSignalNoiseMargin - The value in dB - Upstream noise margin 6 - 31
- TransmissionMode - Possible values are “Auto Mode”, “ANSI”, “G.DMT”. Default value is the ‘Auto Mode’.

Note: The following data will be specified for each of the 8 cross connections that need to be configured. Typically, a single cross connection is configured per subscriber. At least one cross connection has to be data filled to add a new subscriber.

b. Required data for each cross connection

- VCC.ID - Virtual Cross Connection Identifier. The value is between 1-8 both inclusive.
 - VPI - Virtual Path Identifier. Range 16 - 31.
 - VCI - Virtual Circuit Identifier. Range 33 - 2047.
 - UpStreamTrafficDescriptor - Name of the traffic descriptor user label for transmission.
 - DownStreamTrafficDescriptor - Name of the traffic descriptor user label for reception.
 - State - State of the cross connection. Possible values are “Inactive”, “Active”.
 - Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - RC - Return Code
 - MsgTxt - Text (string) detailing specific results of the operation in terms of Success or Failure of the operation and possible indication of failure reason.
- 3. AddCrossConnection** - Interface/Method that will add a new subscriber with the specified cross connection data. This method has the following parameters:

Note: Typically, a single cross connection is configured per subscriber.

- Input data:
 - usn
 - version
 - VMGName
 - TerminationPoint
 - VCC.ID - Virtual Cross Connection Identifier. The value range is 1-8 both inclusive.
 - VPI - Virtual Path Identifier. Range 16 - 31.
 - VCI - Virtual Circuit Identifier. Range 33 - 2047.
 - UpStreamTrafficDescriptor - Name of the traffic descriptor user label for transmission.
 - DownStreamTrafficDescriptor - Name of the traffic descriptor user label for reception.
 - State - State of the cross connection. Possible values are “Inactive”, “Active”.
- Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - RC - Return Code
 - MsgTxt - Text (string) detailing specific results of the operation in terms of Success or Failure of the operation and possible indication of failure reason.
- 4. **ModifySubscriber** - Interface/Method that will modify an existing subscriber with the given configuration data without having to delete and re-add the service. Using this method, multiple cross connection data can also be modified in one transaction. This method has the following parameters:
 - Input data:
 - usn
 - version
 - VMGName
 - TerminationPoint

Note: Zero or more of the following config parameters that need to be modified are specified as the input data. This configuration data applies to all cross connections in the circuit.

- DownMaxSpeed - The value in kbits, Max DownStream 32 - 13376, multiple of 32
- UpMaxSpeed - The value in kbits, Max Upstream 32 - 1440, multiple of 32.
- DownMaxInterleaveDelay - The value in milliseconds - DownStream delay 10 - 255
- UpMaxInterleaveDelay - The value in milliseconds - Upstream delay 10 - 255
- DownSignalNoiseMargin - The value in dB - DownStream noise margin 6 - 31
- UpSignalNoiseMargin - The value in dB - Upstream noise margin 6 - 31
- TransmissionMode - Possible values are “Auto Mode”, “ANSI”, “G.DMT”.

Note: The following data will be specified for each of the 8 cross connections that need to be modified.

Only the data that needs to be modified is specified for each cross connection. Zero or more of the following are specified as input data for each cross connection. For any cross connection data that needs to be modified, the VCC.ID is the required parameter to identify the VCC.

- VCC.ID - Virtual Cross Connection Identifier. The value is between 1-8 both inclusive.
- VPI - Virtual Path Identifier. Range 16 - 31.
- VCI - Virtual Circuit Identifier. Range 33 - 2047.
- UpStreamTrafficDescriptor - Name of the traffic descriptor user label for transmission.
- DownStreamTrafficDescriptor - Name of the traffic descriptor user label for reception.
- State - State of the cross connection. Possible values are “Inactive”, “Active”.
- Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - RC - Return Code

- MsgTxt - Text (string) detailing specific results of the operation in terms of Success or Failure of the operation and possible indication of failure reason.

5. ModifyCrossConnection - Interface/Method that will modify an existing subscriber with the given cross connection data, without having to delete and re-add the service. This method has the following parameters:

- Input data:

- usn
- version
- VMGName
- TerminationPoint
- VCC.ID - Virtual Cross Connection Identifier. The value is between 1-8 both inclusive.

Note: Only the data that needs to be modified is specified for the cross connection. One or more of the following are specified as input data.

- VPI - Virtual Path Identifier. Range 16 - 31.
- VCI - Virtual Circuit Identifier. Range 33 - 2047.
- UpStreamTrafficDescriptor - Name of the traffic descriptor user label for transmission.
- DownStreamTrafficDescriptor - Name of the traffic descriptor user label for reception.
- State - State of the cross connection. Possible values are “Inactive”, “Active”.

- Output data:

- usn (value should be the same as the input)
- version (value should be the same as the input)
- RC - Return Code
- MsgTxt - Text (string) detailing specific results of the operation in terms of Success or Failure of the operation and possible indication of failure reason.

6. DeleteSubscriber - Interface/Method that deletes the specified subscriber data.

Note: All the cross connection data and the circuit configuration data are deleted. To add a new subscriber for this circuit will require specifying the optional configuration parameters as well as the cross connection parameters using the 'AddSubscriber' method.

This method has the following parameters:

- Input data:
 - usn
 - version
 - VMGName
 - TerminationPoint
 - Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - RC - Return Code
 - MsgTxt - Text (string) detailing specific results of the operation in terms of Success or Failure of the operation and possible indication of failure reason.
7. **DeleteCrossConnection** - Interface/Method that deletes the specified cross connection data.

Note: All the cross connection data is deleted. To add cross connection back, use the 'AddCrossConnection' method.

This method has the following parameters:

- Input data:
 - usn
 - version
 - VMGName
 - TerminationPoint
 - VCC.ID - Virtual Cross Connection Identifier. The value is between 1-8 both inclusive.
- Output data:
 - usn (value should be the same as the input)
 - version (value should be the same as the input)
 - RC - Return Code

- MsgTxt - Text (string) detailing specific results of the operation in terms of Success or Failure of the operation and possible indication of failure reason.

11.4 XML Commands

The following are examples of XML commands and the corresponding response messages for each operation supported for ADSL flow through provisioning. The XML coding in these examples is formatted for ease of understanding.

Figure 200 An example of GetSubscriber request:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CommandList>
  <Command>
    <Interface>AdslCircuitProvisioningManager</Interface>
    <Methods>
      <GetSubscriber usn="1" version="1.0">
        <Parameters>
          <VMGName>LAKE017-6-2</VMGName>
          <TerminationPoint>tp/02/03</TerminationPoint>
        </Parameters>
      </GetSubscriber>
    </Methods>
  </Command>
</CommandList>
```

Figure 201 An example of **GetSubscriber** response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CommandList>
  <Response>
    <Interface>AdslCircuitProvisioningManager</Interface>
    <Methods>
      <GetSubscriber usn="1" version="1.0">
        <ReturnData>
          <VMGName>LAKE017-6-2</VMGName>
          <TerminationPoint>tp/02/03</TerminationPoint>
          <UpMaxSpeed> 1440</UpMaxSpeed>
          <DownMaxSpeed>13376</DownMaxSpeed>
          <UpSignalNoiseMargin>6</UpSignalNoiseMargin>
          <DownSignalNoiseMargin>6</DownSignalNoiseMargin>
          <UpMaxInterleaveDelay>10</UpMaxInterleaveDelay>
          <DownMaxInterleaveDelay>10</DownMaxInterleaveDelay>
          <TransmissionMode>Auto Mode</TransmissionMode>
          <VCC
            ID="1"
            VPI="0"
            VCI="0"
            UpStreamTrafficDescriptor="userTDLabel1"
            DownStreamTrafficDescriptor="userTDLabel1"
            State="Active"
          />
          <VCC
            ID="2"
            VPI="1"
            VCI="2"
            UpStreamTrafficDescriptor="userTDLabel2"
            DownStreamTrafficDescriptor="userTDLabel2"
            State="Active"
          />
          <RC>0</RC>
          <MsgTxt>GetSubscriber operation was successful.</MsgTxt>
        </ReturnData>
      </GetSubscriber>
    </Methods>
  </Response>
</CommandList>
```

AddSubscriber Command

Figure 202 An example of AddSubscriber (with 2 cross connections) request:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CommandList>
  <Command>
    <Interface>AdslCircuitProvisioningManager</Interface>
    <Methods>
      <AddSubscriber usn="1" version="1.0">
        <Parameters>
          <VMGName>LAKE017-6-2</VMGName>
          <TerminationPoint>tp/02/03</TerminationPoint>
          <UpMaxSpeed> 1440</UpMaxSpeed>
          <DownMaxSpeed>13376</DownMaxSpeed>
          <UpMaxInterleaveDelay>10</UpMaxInterleaveDelay>
          <DownMaxInterleaveDelay>10</DownMaxInterleaveDelay>
          <UpSignalNoiseMargin>6</UpSignalNoiseMargin>
          <DownSignalNoiseMargin>6</DownSignalNoiseMargin>
          <TransmissionMode>Auto Mode</TransmissionMode>
          <VCC>
            <ID>1</ID>
            <VPI>21</VPI>
            <VCI>40</VCI>
            <UpStreamTrafficDescriptor>upTD</UpStreamTrafficDescriptor>
            <DownStreamTrafficDescriptor>dnTD</DownStreamTrafficDescriptor>
            <State>Active</State>
          </VCC>
          <VCC>
            <ID>2</ID>
            <VPI>21</VPI>
            <VCI>400</VCI>
            <UpStreamTrafficDescriptor>upTD</UpStreamTrafficDescriptor>
            <DownStreamTrafficDescriptor>dnTD</DownStreamTrafficDescriptor>
            <State>Inactive</State>
          </VCC>
        </Parameters>
      </AddSubscriber>
    </Methods>
  </Command>
</CommandList>
```

Figure 203 An example of AddSubscriber response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CommandList>
  <Response>
    <Interface>AdslCircuitProvisioningManager</Interface>
    <Methods>
      <AddSubscriber usn="1" version="1.0">
        <ReturnData>
          <RC>0</RC>
          <MsgTxt>AddSubscriber operation was successful.
          </MsgTxt>
        </ReturnData>
      </AddSubscriber>
    </Methods>
  </Response>
</CommandList>
```

AddCrossConnection Command

Figure 204 An example of AddCrossConnection request:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CommandList>
  <Command>
    <Interface>AdslCircuitProvisioningManager</Interface>
    <Methods>
      <AddCrossConnection usn="1" version="1.0">
        <Parameters>
          <VMGName>LAKE017-6-2</VMGName>
          <TerminationPoint>tp/02/03</TerminationPoint>
          <VCC>
            <ID>8</ID>
            <VPI>21</VPI>
            <VCI>500</VCI>
          </VCC>
          <UpStreamTrafficDescriptor>upTD</UpStreamTrafficDescriptor>
          <DownStreamTrafficDescriptor>dnTD</DownStreamTrafficDescriptor>
          <State>Active</State>
        </Parameters>
      </AddCrossConnection>
    </Methods>
  </Command>
</CommandList>
```

Figure 205 An example of AddCrossConnection response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CommandList>
  <Response>
    <Interface>AdslCircuitProvisioningManager</Interface>
    <Methods>
      <AddCrossConnection usn="1" version="1.0">
        <ReturnData>
          <RC>0</RC>
        </ReturnData>
        <MsgTxt>AddCrossConnection operation was successful.</MsgTxt>
      </AddCrossConnection>
    </Methods>
  </Response>
</CommandList>
```

ModifySubscriber Command:

Figure 206 An example of ModifySubscriber request:

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CommandList>
  <Command>
    <Interface>AdslCircuitProvisioningManager</Interface>
    <Methods>
      <ModifySubscriber usn="1" version="1.0">
        <Parameters>
          <VMGName>LAKE017-6-2</VMGName>
          <TerminationPoint>tp/02/03</TerminationPoint>
          <UpMaxInterleaveDelay>15</UpMaxInterleaveDelay>
          <DownSignalNoiseMargin>8</DownSignalNoiseMargin>
          <VCC>
            <ID>1</ID>
            <VPI>31</VPI>
            <VCI>2047</VCI>
            <State>Inactive</State>
          </VCC>
          <VCC>
            <ID>2</ID>
            <State>Active</State>
          </VCC>
        </Parameters>
      </ModifySubscriber>
    </Methods>
  </Command>
</CommandList>

```

Figure 207 An example of **ModifySubscriber** response:

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CommandList>
  <Response>
    <Interface>AdslCircuitProvisioningManager</Interface>
    <Methods>
      <ModifySubscriber usn="1" version="1.0">
        <ReturnData>
          <RC>0</RC>
          <MsgTxt>ModifySubscriber operation was
successful.</MsgTxt>
        </ReturnData>
      </ModifySubscriber>
    </Methods>
  </Response>
</CommandList>

```

ModifyCrossConnection Command

Figure 208 An example of ModifyCrossConnection request:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CommandList>
  <Command>
    <Interface>AdslCircuitProvisioningManager</Interface>
    <Methods>
      <ModifyCrossConnection usn="1" version="1.0">
        <Parameters>
          <VMGName>LAKE017-6-2</VMGName>
          <TerminationPoint>tp/02/03</TerminationPoint>
          <VCC>
            <ID>8</ID>
            <VPI>30</VPI>
            <VCI>300</VCI>
            <State>Active</State>
          </VCC>
        </Parameters>
      </ModifyCrossConnection>
    </Methods>
  </Command>
</CommandList>
```

Figure 209 An example of **ModifyCrossConnection** response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CommandList>
  <Response>
    <Interface>AdslCircuitProvisioningManager</Interface>
    <Methods>
      <ModifyCrossConnection usn="1" version="1.0">
        <ReturnData>
          <RC>0</RC>
        </ReturnData>
        <MsgTxt>ModifyCrossConnection operation was successful.</MsgTxt>
      </ModifyCrossConnection>
    </Methods>
  </Response>
</CommandList>
```

DeleteSubscriber Command:Figure 210 An example of **DeleteSubscriber** request:

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CommandList >
  <Command>
    <Interface>AdslCircuitProvisioningManager</Interface>
    <Methods>
      <DeleteSubscriber usn="1" version="1.0">
        <Parameters>
          <VMGName>LAKE017-6-2</VMGName>
          <TerminationPoint>tp/02/03</TerminationPoint>
        </Parameters>
      </DeleteSubscriber>
    </Methods>
  </Command>
</CommandList>

```

Figure 211 An example of **DeleteSubscriber** response:

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CommandList>
  <Response>
    <Interface>AdslCircuitProvisioningManager</Interface>
    <Methods>
      <DeleteSubscriber usn="1" version="1.0">
        <ReturnData>
          <RC>0</RC>
          <MsgTxt>DeleteSubscriber operation wa
successful.</MsgTxt>
        </ReturnData>
      </Methods>
    </Response>
  </CommandList>

```


DeleteCrossConnection Command

Figure 212 An example of **DeleteCrossConnection** request:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CommandList>
  <Command>
    <Interface>AdslCircuitProvisioningManager</Interface>
    <Methods>
      <DeleteCrossConnection usn="1" version="1.0">
        <Parameters>
          <VMGName>LAKE017-6-2</VMGName>
          <TerminationPoint>tp/02/03</TerminationPoint>
          <VCC>
            <ID>1</ID>
          </VCC>
        </Parameters>
      </DeleteCrossConnection>
    </Methods>
  </Command>
</CommandList>
```

Figure 213 An example of **DeleteCrossConnection** response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CommandList>
  <Response>
    <Interface>AdslCircuitProvisioningManager</Interface>
    <Methods>
      <DeleteCrossConnection usn="1" version="1.0">
        <ReturnData>
          <RC>0</RC>
          <MsgTxt>DeleteCrossConnection operation was
successful.</MsgTxt>
        </ReturnData>
      </DeleteCrossConnection>
    </Methods>
  </Response>
</CommandList>
```

11.5 User Authorization for ADSL Provisioning operations

In addition to users belonging to “succssn” group to login to OSSGate, user needs to be in application specific groups to perform specific operations. Each operation is associated with one or more user groups. To execute a command, user must belong to at least one of the associated user groups. The user groups associated with ADSL flow through provisioning operations are detailed in Table-14 below.

Table 17: ADSL Commands Authorization levels

Command	User	Group			
	Inadm	Inrw	Inmtc	Insprov	Inro
getSubscriber	X	X	X	X	X
addSubscriber	X	X		X	
addCrossConnection	X	X		X	
modifySubscriber	X	X		X	
modifyCrossConnection	X	X		X	
deleteSubscriber	X	X		X	
deleteCrossConnection	X	X		X	

Figure 214An example of authorization failure response

```

<?xml version="1.0" encoding="UTF-8"?>
<CommandList>
  <Response>
    <Interface>AdslCircuitProvisioningManager</Interface>
    <Methods>
      <GetSubscriber usn="1" version="1.0">
        <RC>501</RC>
        <MsgTxt>Insufficient security privileges to perform this action</MsgTxt>
      </GetSubscriber>
    </Methods>
  </Response>
</CommandList>

```

11.6 Return Codes

Table 18: Return Code Definitions

Return Code	MsgText
0	Request Successful.
1	Operation failure. The name of ADSL circuit received is invalid or the circuit can not be identified.
2	Add Operation Failure. ATM index not found in interface MIB table
3	Modify Operation Failure. ATM index not found in interface MIB table
4	Delete Operation Failure. ATM index not found in interface MIB table
5	Query request failed.
501	Insufficient security privileges to perform this action.

11.7 Limitations and Restrictions

1. Endpoint names must include the VMG name and the termination point.
2. The VMG name must conform to the following format:
SITEXXX-Y-Z - Example: LAKE017-6-2
XXX is the frame number within the office (000 to 511 both inclusive)
Y is the logical frame number within the physical MG (0 to 7 both inclusive)
Z is the shelf number (0 to 3 both inclusive)
3. The termination point must conform to the following format:
tp/AA/BB - Example tp/02/04
AA is the card number (02-09 (both inclusive) and 14-21 (both inclusive))
BB is circuit number (00 to 31 both inclusive)
4. Abort operation is not supported. The ability to abort an XML request received from OSSGate is not supported.

12: Batch Provisioning Tool

The Batch Provisioning Tool (BPT) is a command line user interface. It uses OSSGate as the backend to perform all batch operations. BPT command line user interface allows users to batch provision line and ADSL circuit commands. Following functionality is supported by BPT.

- Execute Line and ADSL batch operation.
- Execute Line endpoint provisioning for third party large line gateways (New in SN08).
- View / Delete Batch operation results
- View / Delete Batch operation logs.

12.1 Pre-requisite to use Batch Provisioning Tool

User must be logged in to the CS2000 management tool server.

User must belong to group “succssn” to start Batch provisioning tool.

User must have appropriate authorization to execute the required line provisioning or ADSL or line endpoint provisioning commands.

Batch provisioning input files must be generated by user before using the tool.

12.2 Using Batch Provisioning Tool

On successful login to the CS2000 management tool server, user can type “bpt” from any directory level. Once successfully authenticated the main menu will be displayed. A typical login session is shown below

Example of BPT login (User input is highlighted in blue):
\$bpt

BPT Main Menu

```
=====
Batch Provisioning Tool (BPT V1.0)
=====
```

Username: tom

Password:

Login in progress...

You are currently logged in as: tom!

=====

Main Menu:

=====

- (1) Execute Batch File
- (2) Display Output
- (3) Display Logs
- (4) Delete Output or Log Files
- (h) Help

- (x) Exit

Selection: [1/2/3/4/h/x:1]

Main Menu:

Option (1) Execute Batch File permits user to execute batch provisioning commands. When option (1) is selected, the “Provisioning Input Entry Menu” will be displayed as shown in 13.2.1.

Option (2) Display Output permits user to view the provisioning output results. The output files are in a text format for line provisioning and in XML format for ADSL provisioning.

Option (3) Display Logs is permits user the view the log files that are associated to the provisioned batch operation.

Option (4) Delete Output or Log Files permits user to remove any unnecessary output or log files.

Option (h) Help provides a generic operation help using BPT.

Option (x) Exit permits user to exit out the BPT.

12.2.1 Provisioning Input Entry Menu:

When Option (1) Execute Batch File of the Main Menu is selected, BPT will prompt the following menu.

Provisioning Input Entry Menu

```
-----
=====
Provisioning Input Entry Menu:
=====
```

- (1) Lines
- (2) ADSL Lines
- (3) Line Endpoints
- (4) Go to shell prompt
- (r) Return to the main menu.
- (x) Exit BPT.

Selection: [1/2/3/4/r/x:1]

```
-----
Option (1) lines permits user to do lines batch provisioning.
Option (2) ADSL permits user to do the ADSL lines batch provisioning.
Option (3) Line Endpoints permits user to do lines endpoint batch provisioning.
```

When the option 3 is selected, BPT will prompt:

```
=====
Line Endpoints Sub-Menu:
=====
```

- (1) Provision third party large line endpoints*
- (2) Export 'Query Line Endpoints' output*
- (b) Back to the previous menu*
- (r) Return to the main menu*
- (x) Exit BPT*

Selection: [1/2/b/r/x:1]1

If "1" is selected from the above "Line Endpoint Sub-Menu:" or if the option 1 or 2 is selected from the "Provisioning Input Entry Menu:" menu, BPT will prompt:

*Which mode would you like to have the log file on?
Critical/Verbose/Major/Minor [c/v/ma/mi:c]:c*

Please enter the input file name:

Note: The data needed for batch provisioning must be generated by user. The input file should be available with proper permissions for use in the CS2000 management tool server. The files can be located in any directory. The absolute path of the input file must be entered.

Sample input file templates are located at:

/opt/nortel/NTsesm/tools/bpt/templates

After the input file name is entered, the tool processes the batch. For example, after entering the line batch provisioning command file "linesBatch", here is what BPT will display to the user:

Please wait while processing the batch commands...

.....

Provisioning is complete! The output file is located at:

/opt/nortel/NTsesm/tools/bpt/output/lines/linesBatch_02100
2_114504.out

Then, BPT will display the Main Menu as in Section 13.2.1 again for further selection.

Option (4) "Go to shell prompt" brings the user to the command line to perform UNIX commands. User can type "exit" to go back the "Provisioning Input Entry Menu".

Option (r) "Return to the main menu" allows the user to go back to the Main Menu in section 13.2

Option (x) "Exit" allows the user to exit out the BPT.

12.2.2 Line Endpoint provisioning Sub-menu:

If Option (3) is selected from the main menu BPT will prompt:

=====

Line Endpoints Sub-Menu:

=====

- (1) Provision third party large line endpoints*
- (2) Export 'Query Line Endpoints' output*
- (b) Back to the previous menu*
- (r) Return to the main menu*
- (x) Exit BPT*

Selection: [1/2/b/r/x:1]2

Option 2 takes the output XML file from one of the queryLineEndpoints or queryAllLineEndpoints command as input and generates an XML suitable for addLineEndpoints operation. The exported file will be available at the following location:

/opt/nortel/NTsesm/tools/bpt/output/export

The generated output file name has the same format as the other commands which is the input file name with appended current time stamp. The contents of the generated file will exactly match the format of the addLineEndpoint XML schema. Please look at addLineEndpoint examples in the earlier section for the details.

12.2.3 Display Output Menu:

When Option (2) Display Output File of the Main Menu is selected, the following "Display Output Menu" will be displayed.

Display Output Menu

=====

Display Output Menu:

=====

- (1) Lines
- (2) ADSL Lines
- (3) Line Endpoints
- (4) Go to shell prompt

- (r) Return to the main menu.
- (x) Exit BPT.

Selection: [1/2/3/4/r/x:1] 3

Option (1) lines permits user to view lines batch provisioning output. Option (2) ADSL permits user to view the ADSL lines batch provisioning output.

Option (3) for Line endpoints will display the following sub-menu

=====

Line Endpoints Sub-Menu:

=====

- (1) Provision third party large line endpoints
- (2) Export 'Query Line Endpoints' output
- (b) Back to the previous menu
- (r) Return to the main menu
- (x) Exit BPT

Selection: [1/2/b/r/x:1] 1

If the user selects Option (1) for lines or Option(2) for ADSL output or any of the sub-options from the above "Line Endpoints Sub-Menu", BPT will display the output files that are currently in the output directory for that selected option. For example, if the user selected option (1) lines, BPT will display all the output files that are currently in the lines output directory. The following example lists the line provisioning output files,

The following files are listed in the output directory:

linesBatch_020930_154219.out

linesBatch_021001_131924.out

linesBatch_021001_111451.out

linesBatch_021002_114504.out

Please enter the file name:

User just needs to enter the file name. BPT will display the file content. Use space bar to view the next page. Type "q" at any time to quit without having to finish viewing the file

After viewing one output file, ability to view other available files exist by going back to the previous menu. For example, the following menu will be displayed after the user finishes viewing a line provisioning output file.

Display Line Output File Menu

=====

Display lines output file menu:

=====

- (1) View another lines output File
- (b) Back to the previous menu.
- (r) Return to the main menu.
- (x) Exit BPT.

Selection: [1/b/r/x:1]

Note: The CS2000 Management tool provides a GUI based application named "Batch configuration Monitor" that displays the ADSL provisioning output files and the "Line Endpoints" provisioning output files, without the XML tags, in a easy readable form.

12.2.4 Display Log File Menu:

If Option (3) Display Log Files is selected from the Main Menu, the following menu will be displayed.

=====
Display Log File Menu:
=====

- (1) Lines
- (2) ADSL
- (3) Line Endpoints
- (4) Go to shell prompt
- (r) Return to the main menu.
- (x) Exit BPT.

Selection: [1/2/3/4/r/x:1]

This menu works exactly like “Display Output Menu” in section 13.2.2. The only difference is that it allows the user to view log files instead of the provisioning output files.

12.2.5 Delete Output or Log Files

When Option (4) Delete Output or Log Files is selected, the following menu will be displayed.

Delete Output or Log File Menu

=====
Delete Files Menu:
=====

- (1) Lines
- (2) ADSL

- (3) Line Endpoints
- (4) Go to shell prompt
- (r) Return to the main menu.
- (x) Exit BPT.

Selection: [1/2/3/4/r/x:1]

Option (1) "lines" permits users to delete the log or output files for lines provisioning.

Option (2) permits users to delete the log or output files for ADSL lines provisioning

Option (3) permits users to delete the log or output files from any of the "Line Endpoints" operations. The "Line Endpoints" has the same sub-menu as in the previous sections to facilitate working with the output or log files for both the provisioning and the export commands. The sub-menu is displayed as follows:

=====

Line Endpoints Sub-Menu:

- =====
- (1) Provision third party large line endpoints
 - (2) Export 'Query Line Endpoints' output
 - (b) Back to the previous menu
 - (r) Return to the main menu
 - (x) Exit BPT

Selection: [1/2/b/r/x:1]

The Option (1) or Option (2) menus from the above sub-menu are displayed as follows:

=====

Delete line_ep File Menu:

=====

- (1) Delete line_ep Output Files
- (2) Delete line_ep Log Files
- (3) Go to shell
- (b) Back to the previous menu.
- (r) Return to the main menu.
- (x) Exit BPT.

Selection: [1/2/3/b/r/x:1]

- (1) Delete export Output Files
- (2) Delete export Log Files
- (3) Go to shell
- (b) Back to the previous menu.
- (r) Return to the main menu.
- (x) Exit BPT.

Selection: [1/2/3/b/r/x:1]

Once the Option (1) or Option (2) is taken from the above sub-menus, the menu flow is the same as in the "lines" or "ADSL" as explained in detail in the later section.

All the commands work in the same way. An example of "lines" command is explained below.

When the Option (1) from the "Delete Files Menu:" is selected, the following menu will be displayed.

12.2.5.1 Delete Lines File Menu

=====

Delete lines File Menu:

=====

- (1) Delete lines Output Files
- (2) Delete lines Log Files
- (3) Go to shell
- (b) Back to the previous menu.
- (r) Return to the main menu.
- (x) Exit BPT.

Selection: [1/2/3/b/r/x:1]

Option (1) permits user to delete output files for the selected provisioning category. Option (2) permits user to delete the log files for the selected provisioning category.

When option (1) or (2) is selected, BPT will display the output files that are in the output directory for that provisioning category. An example listing when "delete lines output files" is selected is listed below

The following files are listed in the specified directory:

linesBatch_020930_154219.out
linesBatch_021001_131924.out

linesBatch_021001_111451.out
linesBatch_021002_114504.out

Please enter the file name(s) that you want to delete (you can enter more than 1 file using the space as a separator:

The user just needs to enter the output file name in order to have it deleted.

It works the same way if the user wants to delete one or more log files. The command returns a confirmation message on files that were deleted.

Note 1: In lines batch, each line represent a SERVORD+ command. This is valid even when the line length exceeds 75 characters. There is no need to add "+" when the commands exceed 75 characters while using BPT

Third party gateway provisioning limitations

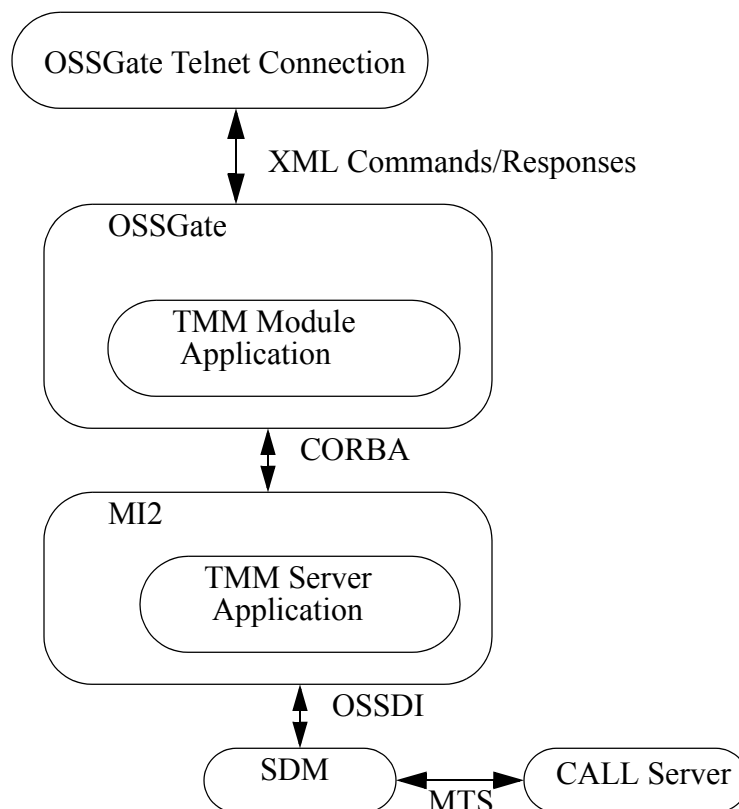
Conversion of existing third party large line GW data to SN08 format is not supported. To use the third party gateway provisioning interface for the existing GW and the associated end-points, service must be de-provisioned and re-added in SN08. Both new and old format data will be allowed to co-exist in SN08, however the existing GW and the end-points will be managed in the same format as defined in the previous release.

The capabilities can not be fully utilized until all the components of the OAMP are at the base SN08 load line up. There is a dependency on the new LGRPTYPE in the SN08 CS2K Core that will be used by the third party GW nodes in the Core. The GWC devices are required to be on the 08 release as well to support spare profiles.

13: Trunk Maintenance with OSSGate

The TMM application enables authenticated OSS clients to perform full maintenance capability for ISUP, PRI, PTS trunks and query all end point states in a trunk gateway. The maintenance operations include QES, POST, BSY, RTS, BSY INB, FRLS trunks in a trunk gateway. Access to this functionality is by connection to OSSGate.

Figure 215 TMM in the Succession Network



13.1 Supported TMM Commands

Following XML commands are supported for TMM using OSSGate.

- PostByGatewayName
- QESByGatewayName

- BSYByGatewayName
- RTSByGatewayName
- INBByGatewayName
- FRLSByGatewayName
- PostByTrunkCLLI
- BSYByTrunkCLLI
- INBByTrunkCLLI
- RTSByTrunkCLLI
- FRLSByTrunkCLLI
- ICOTTest
- PostGroupDChannelByTrunkCLLI
- RTSDChannelByTid
- INBDChannelByTid
- BSYDChannelByTid
- PostByCarrier
- QESByCarrier
- BSYByCarrier
- INBByCarrier
- RTSByCarrier
- FRLSByCarrier
- GetTrunkCllisByGatewayName
- GetGatewayNames
- GetCarriers
- GetCMCLLIs
- GetGWCNames

13.2 Terminology, Description of Method Parameters

- **GatewayName** - Name of the media gateway. TYPE *string*. Format: Min length 1; Max length 32
- **EndpointRange** - Range of maintained endpoints. TYPE *string*. Format: [Start member number [-End member number]]* [Start member number [-[End member number]]]
Examples:
a. "1" - Post trunk 1

- b. "1,2" - Post trunk 1 and 2
- c. "1-3" - Post trunks 1 through 3 inclusive
- d. "1, 3-5" - Post trunks 1 and 3-5 inclusive
- e. "1, 3-" - Post trunks 1, and post the rest of the trunk group starting with trunk member 3.
- **TrunkMembers** - Range of maintained trunk members. TYPE *string*. Format: same as EndpointRange
- **CMCLLI** - CLLI Name of Call Server. TYPE *string*. Format: Min length 1; Max length 32
- **TrunkCLLI** - CLLI Name of trunks. TYPE *string*. Format: Min length 1; Max length 16
- **GWCName** - Name of the gateway controller. TYPE *string*. Format: Min length 1; Max length 32
- **CarrierNames** - Name of the carriers. TYPE *string*. Format: Min length 1
- **ShowDetails** - Whether show the details of trunks. TYPE *boolean*.
- **FilterState** - The valid states of trunks. TYPE *string*. The valid values include "ALL", "CPD", "IDL", "MB", "NEQ", "INB", "NMB", "PMB", "RMB", "SB", "CPB", "CFL", "LO", "DEL", "INI", "RES", "SZD", "DMB", "DFL", "INS", "STB" and "UNKNOWN".
- **NodeNumber** - The Node No. TYPE *string*.
- **Value** - The value of trunks' state. TYPE *string*. The valid values are the same as FilterState except "ALL".
- **Count** - The number of trunks in the state. TYPE *non negative integer*.
- **State** - The current state for the trunk. TYPE *string*. Min length 2, Max length 5. The valid values are the same as FilterState except "ALL".
- **ConnectedTo** - What the trunk is connected to. TYPE *string*.
- **TrunkDirection** - The trunk direction. TYPE *string*.
- **TrunkSignaling** - The trunk signaling. TYPE *string*.
- **PMType** - The PM type. TYPE *string*. The valid values include "PRI", "ISUP", and "PTS".
- **PMNumber** - The PM number. TYPE *string*.
- **TerminalNumber** - Terminal Number. TYPE *string*.

- **EndpointName** - Name of endpoint. TYPE *string*.
- **TrunkMember** - Trunk member. TYPE *non negative integer*.
- **PMCarrier** - The PM Carrier. TYPE *string*.
- **PMTimeSlot** - The PM timeslot. TYPE *string*.
- **TrunkType** - Type of Trunk. TYPE *string*.
- **TestResult** - ICOT test result. TYPE *string*.
- **ContinuityCondition** - The continuity condition. TYPE *string*.
- **AdditionalInfo** - The additional information. TYPE *string*.
- **CallID** - The call identification. TYPE *string*.
- **SupplementInfo** - The supplement information. TYPE *string*.
- **Version** - The version refers to the version of the method or operation - e.g., the PostByGatewayName operation may have version 1.0 and 2.0, where version 1.0 takes parameters X and Y and version 2.0 takes parms X, Y, and Z. TYPE *string*.

The following parameters will be returned when some errors happen.

- **Number** - No. of the error. TYPE *string*. Required
- **Message** - Message of the error. TYPE *string*. Required
- **Severity** - Type of the error. TYPE *string*. The valid values include "INFORMATION", "WARNING", "MINOR", "MAJOR" and "CRITICAL". Optional
- **Param1** - First parameter. TYPE *string*. Optional
- **Param2** - Second parameter. TYPE *string*. Optional
- **Param3** - Third parameter. TYPE *string*. Optional

13.3 Description of Method Parameters

1. PostByGatewayName - Interface/Method that gets all the endpoint information for ISUP, PRI, PTS trunks based on Gateway Name and EndpointRange.
 - Input data:
 - Version - Optional
 - GatewayName - Required

- EndpointRange - Required
- FilterState - Optional
- ShowDetails - Optional
- Output data:
 - Version - Optional
 - GatewayName - Required
 - NodeNumber - Required
 - FilterState - Required
 - EndpointRange - Required
 - Value - Required
 - Count - Required
 - SupplementInfo - Optional
 - State - Optional
 - ConnectedTo - Optional
 - TrunkDirection - Optional
 - TrunkSignaling -Optional
 - PMType - Optional
 - PMNumber - Optional
 - TerminalNumber - Optional
 - EndpointName - Optional
 - TrunkCLLI - Optional
 - TrunkMember - Optional
 - PMCarrier - Optional
 - PMTimeSlot - Optional
 - TrunkType - Optional
- 2. QESByGatewayName - Interface/Method that query the endpoint state for ISUP, PRI, PTS trunks based on Gateway Name and EndpointRange.
 - Input data:
 - Version - Optional
 - GatewayName - Required
 - EndpointRange - Required
 - FilterState - Optional

- ShowDetails - Optional
- Output data:
- Version - Optional
- GatewayName - Required
- NodeNumber - Required
- FilterState - Required
- EndpointRange - Required
- Value - Required
- Count - Required
- SupplementInfo - Optional
- State - Optional
- TerminalNumber - Optional
- 3. BSYByGatewayName - Interface/Method that will busy ISUP, PRI, PTS trunks based on Gateway Name and EndpointRange.
 - Input data:
 - Version - Optional
 - GatewayName - Required
 - EndpointRange - Required
 - Output data:
 - Version - Optional
 - GatewayName - Required
 - NodeNumber - Required
 - EndpointRange - Required
 - TerminalNumber - Optional
- 4. RTSByGatewayName - Interface/Method that will return ISUP, PRI, PTS trunks to service based on Gateway Name and EndpointRange.
 - Input data:
 - Version - Optional
 - GatewayName - Required.
 - EndpointRange - Required
 - Output data:

- Version - Optional
 - GatewayName - Required
 - NodeNumber - Required
 - EndpointRange - Required
 - TerminalNumber - Optional
5. INBByGatewayName - Interface/Method that will make ISUP, PRI, PTS trunks be in Installation busy state based on Gateway Name and EndpointRange.
- Input data:
 - Version - Optional
 - GatewayName - Required
 - EndpointRange - Required
 - Output data:
 - Version - Optional
 - GatewayName - Required
 - NodeNumber - Required
 - EndpointRange - Required
 - TerminalNumber - Optional
6. FRLSByGatewayName - Interface/Method that will force to release endpoints of ISUP, PRI, PTS trunks based on Gateway Name and EndpointRange.
- Input data:
 - Version - Optional
 - GatewayName - Required
 - EndpointRange - Required
 - Output data:
 - Version - Optional
 - GatewayName - Required
 - NodeNumber - Required
 - EndpointRange - Required
 - TerminalNumber - Optional
7. PostByTrunkCLLI - Interface/Method that gets all the endpoint information for ISUP, PRI, PTS trunks based on Tunk CLLI and Trunk members.

- Input data:
 - Version - Optional
 - CMCLli - Optional
 - TrunkCLli - Required
 - TrunkMembers - Required
- Output data:
 - Version - Optional
 - TrunkCLLI - Required
 - CMCLLI - Required
 - FirstMember - Optional
 - GroupSize - Optional
 - TrunkMembers - Optional
 - State - Optional
 - ConnectedTo - Optional
 - TrunkDirection - Optional
 - TrunkSignaling - Optional
 - PMType - Optional
 - PMNumber - Optional
 - TerminalNumber - Optional
 - EndpointName - Optional
 - NodeNumber - Optional
 - GatewayName - Optional
 - TrunkMember - Optional, TYPE *non negative integer*.
 - PMCarrier - Optional
 - PMTimeSlot - Optional
- 8. BSYByTrunkCLLI - Interface/Method that will busy ISUP, PRI, PTS trunks based on TrunkCLLI and TrunkMembers.
 - Input data:
 - Version - Optional
 - CMCLli - Optional
 - TrunkCLli - Required
 - TrunkMembers - Required

- Output data:
 - Version - Optional
 - TrunkCLLI - Required
 - CMCLLI - Required
 - FirstMember - Optional
 - GroupSize - Optional
 - TrunkMembers - Optional
 - TrunkMember - Optional. TYPE *string*. Its format is the same as TrunkMembers.
- 9. RTSByTrunkCLLI - Interface/Method that will return ISUP, PRI, PTS trunks to service based on TrunkCLLI and TrunkMembers.
 - Input data:
 - Version - Optional
 - CMCLli - Optional
 - TrunkClli - Required
 - TrunkMembers - Required
 - Output data:
 - Version- Optional
 - TrunkCLLI - Required
 - CMCLLI - Required
 - FirstMember - Optional
 - GroupSize - Optional
 - TrunkMembers - Optional
 - TrunkMember - Optional. TYPE *string*. Its format is the same as TrunkMembers.
- 10. INBByTrunkCLLI - Interface/Method that will make ISUP, PRI, PTS trunks be in Installation Busy based on TrunkCLLI and TrunkMembers.
 - Input data:
 - Version - Optional
 - CMCLli - Optional
 - TrunkClli - Required
 - TrunkMembers - Required

- Output data:
 - Version - Optional
 - TrunkCLLI - Required
 - CMCLLI - Required
 - FirstMember - Optional
 - GroupSize - Optional
 - TrunkMembers - Optional
 - TrunkMember - Optional. TYPE *string*. Its format is the same as TrunkMembers.
- 11. FRLSByTrunkCLLI - Interface/Method that will force to release endpoints of ISUP, PRI, PTS trunks based on TrunkCLLI and TrunkMembers.
 - Input data:
 - Version - Optional
 - CMCLli - Optional
 - TrunkClli - Required
 - TrunkMembers - Required. TYPE *string*. Format: member number [, member number]*
 - Examples:
 - a. "1" - Post trunk 1
 - b. "1,2" - Post trunk 1 and 2
 - Output data:
 - Version - Optional
 - TrunkCLLI - Required
 - CMCLLI - Required
 - FirstMember - Optional
 - GroupSize - Optional
 - TrunkMembers - Optional
 - TrunkMember - Optional. TYPE *string*. Its format is the same as TrunkMembers.
- 12. ICOTTest - Interface/Method that will do ICOT test for ISUP trunks based on TrunkCLLI and TrunkMember.
 - Input data:
 - Version - Optional

- CMCLli - Optional
- TrunkCLli - Required
- TrunkMember - Required. TYPE *non negative integer*.
- Output data:
 - Version - Optional
 - TrunkCLLI - Required
 - CMCLLI - Required
 - FirstMember - Optional
 - GroupSize - Optional
 - TrunkMembers - Optional
 - TrunkMember - Optional. TYPE *non negative integer*.
 - TestResult - Optional
 - ContinuityCondition - Optional
 - AdditionalInfo - Optional
 - CallID - Optional

13. PostGroupDChannelByTrunkCLLI - Interface/Method that gets all the endpoint information for PRI DChannel based on TrunkCLLI.

- Input data:
 - Version - Optional
 - CMCLli - Optional
 - TrunkCLli - Required
- Output data:
 - Version - Optional
 - TrunkCLLI - Required
 - CMCLLI - Required
 - FirstMember - Optional
 - GroupSize - Optional
 - TrunkMembers - Optional
 - State - Optional
 - ConnectedTo - Optional
 - TrunkDirection - Optional
 - TrunkSignaling - Optional

- PMType - Optional
 - PMNumber - Optional
 - TerminalNumber - Optional
 - EndpointName - Optional
 - NodeNumber - Optional
 - GatewayName - Optional
- 14.** BSYDChannelByTid - Interface/Method that busy PRI DChannel based on TerminalNumber.
- Input data:
 - Version - Optional
 - CMCLli - Optional.
 - NodeNumber - Required
 - TerminalNumber - Required
 - Force - Optional. If the DChannel is in INS and Force is false, BSYDChannelByTid will fail.
 - Output data:
 - Version - Optional
 - CMCLLI - Required
 - TrunkMember - Optional. TYPE *string*. Its format is the same as TrunkMembers.
- 15.** RTSDChannelByTid - Interface/Method that return PRI DChannel service based on TerminalNumber.
- Input data:
 - Version - Optional
 - CMCLli - Optional.
 - NodeNumber - Required
 - TerminalNumber - Required
 - Output data:
 - Version - Optional
 - CMCLLI - Required
 - TrunkMember - Optional. TYPE *string*. Its format is the same as TrunkMembers.
- 16.** INBDChannelByTid - Interface/Method that make PRI DChannel in installation busy based on TerminalNumber.

- Input data:
 - Version - Optional
 - CMCLli - Optional.
 - NodeNumber - Required
 - TerminalNumber - Required
 - Output data:
 - Version - Optional
 - CMCLLI - Required
 - TrunkMember - Optional. TYPE *string*. Its format is the same as TrunkMembers.
- 17.** PostByCarrier - Interface/Method that gets all the endpoint information for ISUP, PRI, PTS trunks based on Gateway Name and CarrierNames.
- Input data:
 - Version - Optional
 - GatewayName - Required
 - GWCName - Required
 - CarrierNames - Required
 - EndpointRange - Required
 - FilterState - Optional
 - ShowDetails - Optional
 - Output data:
 - Version - Optional
 - GatewayName - Required
 - NodeNumber - Required
 - FilterState - Required
 - EndpointRange - Required
 - Value - Required
 - Count - Required
 - SupplementInfo - Optional
 - State - Optional
 - ConnectedTo - Optional
 - TrunkDirection - Optional

- TrunkSignaling - Optional
- PMType - Optional
- PMNumber - Optional
- TerminalNumber - Optional
- EndpointName - Optional
- TrunkCLLI - Optional
- TrunkMember - Optional
- PMCarrier - Optional
- PMTimeSlot - Optional
- TrunkType - Optional

18. QESByCarrier - Interface/Method that query the endpoint state for ISUP, PRI, PTS trunks based on Gateway Name and Carrier.

- Input data:

- Version - Optional
- GatewayName - Required
- GWCName - Required
- CarrierNames - Required
- EndpointRange - Required
- FilterState - Optional
- ShowDetails - Optional

- Output data:

- Version - Optional
- GatewayName - Required
- NodeNumber - Required
- FilterState - Required
- EndpointRange - Required
- Value - Required
- Count - Required
- SupplementInfo - Optional
- State - Optional
- TerminalNumber - Optional

19. BSYByCarrier - Interface/Method that will busy ISUP, PRI, PTS trunks based on Gateway Name and CarrierNames.
 - Input data:
 - Version - Optional
 - GatewayName - Required
 - GWCName - Required
 - CarrierNames - Required
 - EndpointRange - Required
 - Output data:
 - Version - Optional
 - GatewayName - Required
 - NodeNumber - Required
 - EndpointRange - Required
 - TerminalNumber - Optional.
20. RTSByCarrier - Interface/Method that will return ISUP, PRI, PTS trunks to service based on Gateway Name and CarrierNames.
 - Input data:
 - Version - Optional
 - GatewayName - Required
 - GWCName - Required
 - CarrierNames - Required
 - EndpointRange - Required
 - Output data:
 - Version - Optional
 - GatewayName - Required
 - NodeNumber - Required
 - EndpointRange - Required
 - TerminalNumber - Optional
21. INBByCarrier - Interface/Method that will make ISUP, PRI, PTS trunks in installation busy based on Gateway Name and CarrierNames.
 - Input data:
 - Version - Optional

- GatewayName - Required
- GWCName - Required
- CarrierNames - Required
- EndpointRange - Required
- Output data:
- Version - Optional
- GatewayName - Required
- NodeNumber - Required
- EndpointRange - Required
- TerminalNumber - Optional

22. FRLSByCarrier - Interface/Method that will force to release endpoints of ISUP, PRI, PTS trunks based on Gateway Name and CarrierNames.

- Input data:
- Version - Optional
- GatewayName - Required
- GWCName - Required
- CarrierNames - Required
- EndpointRange - Required
- Output data:
- Version - Optional
- GatewayName - Required
- NodeNumber - Required
- EndpointRange - Required
- TerminalNumber - Optional

23. GetTrunkCllisByGatewayName - Interface/Method that will get trunk CLLIs of ISUP, PRI, PTS trunks based on Gateway Name.

- Input data:
- Version - Optional
- CMCLLI - Optional
- GatewayName - Required
- Output data:

- Version - Optional
 - TrunkClli - Required
- 24.** GetGatewayNames - Interface/Method that will get gateways' names.
- Input data:
 - Version - Optional
 - MaxReturn - Optional.
 - StartingIndex - Optional
 - CMCLLI - Optional
 - GWC - Optional
 - Output data:
 - Version - Optional
 - Names - Required
- 25.** GetCarriers - Interface/Method that will get carriers' names.
- Input data:
 - Version - Optional
 - MaxReturn - Optional.
 - StartingIndex - Optional
 - CMCLLI - Optional
 - GWC - Optional
 - Output data:
 - Version - Optional
 - GatewayName - Required
 - GWCName - Required
 - CarrierNames - Required
- 26.** GetCMCLLIs - Interface/Method that will get CM CLLIs.
- Input data:
 - Version - Optional
 - MaxReturn - Optional.
 - StartingIndex - Optional
 - Output data:
 - Version - Optional

- Names - Required
- 27. GetGWCNames - Interface/Method that will get Gateway Controllers' names.
 - Input data:
 - Version - Optional
 - MaxReturn - Optional.
 - StartingIndex - Optional
 - CMCLLI - Optional
 - Output data:
 - Version - Optional
 - Names - Required

13.4 XML Commands

The following are examples of XML commands and the corresponding response messages for each operation supported for TMM. The XML coding in these examples is formatted for ease of understanding.

Figure 216 An example of **PostByGatewayName** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <PostByGatewayName
        GatewayName="PRIFORDOC"
        EndpointRange="35-36"
        ShowDetails="true"
        FilterState="ALL"
      />
    </Methods>
  </Command>
</CommandList>
```

Figure 217 An example of **PostByGatewayName** response:

```

<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <PostByGatewayName>
        <Header>
          GatewayName="PRIFORDOC"
          EndpointRange="35-36"
          NodeNumber="67"
          FilterState="ALL"
        >
          <Summary>
            <State Value="INB " Count="2"/>
          </Summary>
        </Header>
      </PostByGatewayName>
    </Methods>
  </Response>
</CommandList>

```

Figure 218 An example of **PostByGatewayName** response:

```

<Members>
  <Member>
    TerminalNumber="35"
    State="INB "
    ConnectedTo="          "
    TrunkDirection="2W"
    TrunkSignaling="ISD ISD "
    PMType="GWC_NODE"
    PMNumber="0"
    EndpointName="DS1_10.4"
    TrunkCLLI="PRIFORUPGRADEDOC"
    TrunkMember="3"
    PMCarrier="0"
    PMTimeSlot="1"
    TrunkType="PRI"
  </Member>
  <Member>
    TerminalNumber="36"
    State="INB "
    ConnectedTo="          "
    TrunkDirection="2W"
    TrunkSignaling="ISD ISD "
    PMType="GWC_NODE"
    PMNumber="0"
    EndpointName="DS1_10.5"
    TrunkCLLI="PRIFORUPGRADEDOC"
    TrunkMember="4"
    PMCarrier="0"
    PMTimeSlot="1"
    TrunkType="PRI"
  </Member>
</Members>
</PostByGatewayName>
</Methods>
</Response>
</CommandList>

```

Figure 219 An example of **QESByGatewayName** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <QESByGatewayName
        GatewayName="PRIFORDOC"
        EndpointRange="35-36"
        ShowDetails="true"
        FilterState="ALL"
      />
    </Methods>
  </Command>
</CommandList>
```

Figure 220 An example of **QESByGatewayName** response:

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <QESByGatewayName>
        <Header
          GatewayName="PRIFORDOC"
          EndpointRange="35-36"
          NodeNumber="67"
          FilterState="ALL"
        >
          <Summary>
            <State Value="INB " Count="2"/>
          </Summary>
        </Header>
        <Members>
          <Member
            TerminalNumber="35"
            State="INB "
          />
          <Member
            TerminalNumber="36"
            State="INB "
          />
        </Members>
      </QESByGatewayName>
    </Methods>
  </Response>
</CommandList>
```

Figure 221 An example of **BSYByGatewayName** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <BSYByGatewayName
        GatewayName="PRIFORDOC"
        EndpointRange="35-36"
      />
    </Methods>
  </Command>
</CommandList>
```

Figure 222 An example of **BSYByGatewayName** response:

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <BSYByGatewayName>
        <Header
          GatewayName="PRIFORDOC"
          EndpointRange="35-36"
          NodeNumber="67"
        />
      </BSYByGatewayName>
    </Methods>
  </Response>
</CommandList>
```

Figure 223 An example of **RTSByGatewayName** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <RTSByGatewayName
        GatewayName="PVG194"
        EndpointRange="674-675"
      />
    </Methods>
  </Command>
</CommandList>
```

Figure 224 An example of **RTSByGatewayName** response:

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <RTSByGatewayName>
        <Header
          GatewayName="PVG194"
          EndpointRange="674-675"
          NodeNumber="16"
        />
      </RTSByGatewayName>
    </Methods>
  </Response>
</CommandList>
```

Figure 225 An example of **INBByGatewayName** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <INBByGatewayName
        GatewayName="PVG194"
        EndpointRange="674-675"
      />
    </Methods>
  </Command>
</CommandList>
```

Figure 226 An example of **INBByGatewayName** response:

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <INBByGatewayName>
        <Header
          GatewayName="PVG194"
          EndpointRange="674-675"
          NodeNumber="16"
        />
      </INBByGatewayName>
    </Methods>
  </Response>
</CommandList>
```

Figure 227 An example of **FRLSByGatewayName** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <FRLSByGatewayName
        GatewayName="PVG194"
        EndpointRange="674,675"
      />
    </Methods>
  </Command>
</CommandList>
```

Figure 228 An example of **FRLSByGatewayName** response:

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <FRLSByGatewayName>
        <Header
          GatewayName="PVG194"
          EndpointRange="674,675"
          NodeNumber="16"
        />
      </FRLSByGatewayName>
    </Methods>
  </Response>
</CommandList>
```

Figure 229 An example of **PostByTrunkCLLI** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <PostByTrunkCLLI
        TrunkClli="P194ITLOOPOG1"
        TrunkMembers="2-3"
      />
    </Methods>
  </Command>
</CommandList>
```

Figure 230 An example of **PostByTrunkCLLI** response:

```

<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <PostByTrunkCLLI>
        <Header>
          TrunkCLLI="P194ITLOOPOG1"
          CMCLLI="RTPS"
          FirstMember="1"
          GroupSize="24"
          TrunkMembers="2-3"
        </Header>
        <Members>
          <Member>
            TrunkMember="2"
            State="IDL "
            ConnectedTo=" "
            TrunkDirection="2W"
            TrunkSignaling="S7 S7"
            PMType="GWC_NODE"
            PMNumber="2"
            PMCarrier="0"
            PMTimeSlot="1"
            GatewayName="PVG194"
            EndpointName="SS_6003_VT15_0143.2"
            NodeNumber="16"
            TerminalNumber="674"
          </Member>
          <Member>
            TrunkMember="3"
            State="IDL "
            ConnectedTo=" "
            TrunkDirection="2W"
            TrunkSignaling="S7 S7 "
            PMType="GWC_NODE"
            PMNumber="2"
            PMCarrier="0"
            PMTimeSlot="1"
            GatewayName="PVG194"
            EndpointName="SS_6003_VT15_0143.3"
            NodeNumber="16"
            TerminalNumber="675"
          </Member>
        </Members>
      </PostByTrunkCLLI>
    </Methods>
  </Response>
</CommandList>

```


Figure 231 An example of **BSYByTrunkCLLI** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <BSYByTrunkCLLI
        TrunkClli="P194ITLOOPOG1"
        TrunkMembers="2-3"
      />
    </Methods>
  </Command>
</CommandList>
```

Figure 232 An example of **BSYByTrunkCLLI** response

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <BSYByTrunkCLLI>
        <Header
          TrunkCLLI="P194ITLOOPOG1"
          CMCLLI="RTPS"
          FirstMember="1"
          GroupSize="24"
          TrunkMembers="2-3"
        />
        <Members>
        </Members>
      </BSYByTrunkCLLI>
    </Methods>
  </Response>
</CommandList>
```

Figure 233 An example of **RTSByTrunkCLLI** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <RTSByTrunkCLLI
        TrunkClli="P194ITLOOPOG1"
        TrunkMembers="2-3"
      />
    </Methods>
  </Command>
</CommandList>
```

Figure 234 An example of **RTSByTrunkCLLI** response

```

<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <RTSByTrunkCLLI>
        <Header
          TrunkCLLI="P194ITLOOPOG1"
          CMCLLI="RTPS"
          FirstMember="1"
          GroupSize="24"
          TrunkMembers="2-3"
        />
        <Members>
        </Members>
      </RTSByTrunkCLLI>
    </Methods>
  </Response>
</CommandList>

```

Figure 235 An example of **INBByTrunkCLLI** request:

```

<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <INBByTrunkCLLI
        TrunkClli="P194ITLOOPOG1"
        TrunkMembers="2-3"
      />
    </Methods>
  </Command>
</CommandList>

```

Figure 236 An example of **INBByTrunkCLLI** response

```

<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <INBByTrunkCLLI>
        <Header
          TrunkCLLI="P194ITLOOPOG1"
          CMCLLI="RTPS"
          FirstMember="1"
          GroupSize="24"
          TrunkMembers="2-3"
        />
        <Members>
        </Members>
      </INBByTrunkCLLI>
    </Methods>
  </Response>
</CommandList>

```

Figure 237 An example of **FRLSByTrunkCLLI** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <FRLSByTrunkCLLI
        TrunkClli="P194ITLOOPOG1"
        TrunkMembers="2,3"
      />
    </Methods>
  </Command>
</CommandList>
```

Figure 238 An example of **FRLSByTrunkCLLI** response

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <FRLSByTrunkCLLI>
        <Header
          TrunkCLLI="P194ITLOOPOG1"
          CMCLLI="RTPS"
          FirstMember="1"
          GroupSize="24"
          TrunkMembers="2,3"
        />
        <Members>
        </Members>
      </FRLSByTrunkCLLI>
    </Methods>
  </Response>
</CommandList>
```

Figure 239 An example of **PostByCarrier** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <PostByCarrier
        GatewayName="PRIFORDOC"
        GWCName="GWC-0"
        CarrierNames="DS1_10"
        EndpointRange="35-36"
      />
    </Methods>
  </Command>
</CommandList>
```

Figure 240 An example of **PostByCarrier** response

```

<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <PostByCarrier>
        <Header
          GatewayName="PRIFORDOC"
          EndpointRange="35-36"
          NodeNumber="67"
          FilterState="All"
        >
          <Summary>
            <State Value="CFL " Count="2"/>
          </Summary>
        </Header>
      </PostByCarrier>
    </Methods>
  </Response>
</CommandList>

```

Figure 241 An example of **QESByCarrier** request:

```

<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <QESByCarrier
        GatewayName="PRIFORDOC"
        GWCName="GWC-0"
        CarrierNames="DS1_10"
        EndpointRange="35-36"
        ShowDetails="true"
        FilterState="ALL"
      />
    </Methods>
  </Command>
</CommandList>

```

Figure 242 An example of **QESByCarrier** response

```

<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <QESByCarrier>
        <Header
          GatewayName="PRIFORDOC"
          EndpointRange="35-36"
          NodeNumber="67"
          FilterState="ALL"
        >
          <Summary>
            <State Value="CFL " Count="2"/>
          </Summary>
        </Header>
        <Members>
          <Member
            TerminalNumber="35"
            State="CFL "
          />
          <Member
            TerminalNumber="36"
            State="CFL "
          />
        </Members>
      </QESByCarrier>
    </Methods>
  </Response>
</CommandList>

```

Figure 243 An example of **BSYByCarrier** request:

```

<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <BSYByCarrier
        GatewayName="PRIFORDOC"
        GWCName="GWC-0"
        CarrierNames="DS1_10"
        EndpointRange="35-36"
      />
    </Methods>
  </Command>
</CommandList>

```

Figure 244 An example of **BSYByCarrier** response

```

<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <BSYByCarrier>
        <Header
          GatewayName="PRIFORDOC"
          EndpointRange="35-36"
          NodeNumber="67"
        />
      </BSYByCarrier>
    </Methods>
  </Response>
</CommandList>

```

Figure 245 An example of **RTSByCarrier** request:

```

<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <RTSByCarrier
        GatewayName="PRIFORDOC"
        GWCName="GWC-0"
        CarrierNames="DS1_10"
        EndpointRange="35-36"
      />
    </Methods>
  </Command>
</CommandList>

```

Figure 246 An example of **RTSByCarrier** response

```

<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <RTSByCarrier>
        <Header
          GatewayName="PRIFORDOC"
          EndpointRange="35-36"
          NodeNumber="67"
        />
      </RTSByCarrier>
    </Methods>
  </Response>
</CommandList>

```

Figure 247 An example of **INBByCarrier** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <INBByCarrier
        GatewayName="PRIFORDOC"
        GWCName="GWC-0"
        CarrierNames="DS1_10"
        EndpointRange="35-36"
      />
    </Methods>
  </Command>
</CommandList>
```

Figure 248 An example of **INBByCarrier** response

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <INBByCarrier>
        <Header
          GatewayName="PRIFORDOC"
          EndpointRange="35-36"
          NodeNumber="67"
        />
      </INBByCarrier>
    </Methods>
  </Response>
</CommandList>
```

Figure 249 An example of **FRLSByCarrier** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <FRLSByCarrier
        GatewayName="PRIFORDOC"
        GWCName="GWC-0"
        CarrierNames="DS1_10"
        EndpointRange="35,36"
      />
    </Methods>
  </Command>
</CommandList>
```

Figure 250 An example of **FRLSByCarrier** response

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <FRLSByCarrier>
        <Header
          GatewayName="PRIFORDOC"
          EndpointRange="35,36"
          NodeNumber="67"
        />
      </FRLSByCarrier>
    </Methods>
  </Response>
</CommandList>
```

Figure 251 An example of **PostGroupDChannelByTrunkCLLI** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <PostGroupDChannelByTrunkCLLI
        TrunkCli="PRIFORUPGRADEDOC"
      />
    </Methods>
  </Command>
</CommandList>
```


Figure 252 An example of **PostGroupDChannelByTrunkCLLI** response

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <PostGroupDChannelByTrunkCLLI>
        <Header
          TrunkCLLI="PRIFORUPGRADED0C"
          CMCLLI="COMPACT5"
        />
        <Members>
          <Member
            State="INB "
            TrunkDirection="2W"
            TrunkSignaling="ISD ISD "
            PMType="GWC_NODE"
            PMNumber="0"
            GatewayName="PRIFORDOC"
            EndpointName="DS1_10.2"
            NodeNumber="67"
            TerminalNumber="33"
          />
          <Member
            State="INB "
            TrunkDirection="2W"
            TrunkSignaling="ISD ISD "
            PMType="GWC_NODE"
            PMNumber="0"
            GatewayName="PRIFORDOC"
            EndpointName="DS1_10.3"
            NodeNumber="67"
            TerminalNumber="34"
          />
        </Members>
      </PostGroupDChannelByTrunkCLLI>
    </Methods>
  </Response>
</CommandList>
```

Figure 253 An example of **ICOTTest** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <ICOTTest
        TrunkCli="SUC101ISUPV2LP"
        TrunkMember="2"
      />
    </Methods>
  </Command>
</CommandList>
```

Figure 254 An example of **ICOTTest** response

```

<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <ICOTTest>
        <Header
          TrunkCLLI="SUC101ISUPV2LP"
          CMCLLI="COMPACT2"
          FirstMember="2"
          GroupSize="2"
          TrunkMembers="2"
        />
        <Members>
          <Member
            TrunkMember="2"
            TestResult="TEST_PASSED"
            ContinuityCondition="
            AdditionalInfo="
            CallID=""
          />
        </Members>
      </ICOTTest>
    </Methods>
  </Response>
</CommandList>

```

Figure 255 An example of **BSYDChannelByTid** request:

```

<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <BSYDChannelByTid
        NodeNumber="18"
        TerminalNumber="543"
        Force="true"
      />
    </Methods>
  </Command>
</CommandList>

```

Figure 256 An example of **BSYDChannelByTid** response

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <BSYDChannelByTid>
        <Header
          CMCLLI="COMPACT2"
        />
        <Members>
        </Members>
      </BSYDChannelByTid>
    </Methods>
  </Response>
</CommandList>
```

Figure 257 An example of **RTSDChannelByTid** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <RTSDChannelByTid
        NodeNumber="18"
        TerminalNumber="543"
      />
    </Methods>
  </Command>
</CommandList>
```

Figure 258 An example of **RTSDChannelByTid** response

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <RTSDChannelByTid>
        <Header
          CMCLLI="COMPACT2"
        />
        <Members>
        </Members>
      </RTSDChannelByTid>
    </Methods>
  </Response>
</CommandList>
```

Figure 259 An example of **INBDChannelByTid** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <INBDChannelByTid
        NodeNumber="18"
        TerminalNumber="543"
      />
    </Methods>
  </Command>
</CommandList>
```

Figure 260 An example of **INBDChannelByTid** response

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <INBDChannelByTid>
        <Header
          CMCLLI="COMPACT2"
        />
        <Members>
        </Members>
      </INBDChannelByTid>
    </Methods>
  </Response>
</CommandList>
```

Figure 261 An example of **GetTrunkCllisByGatewayName** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <GetTrunkCllisByGatewayName
        GatewayName="PRIFORDOC"
      />
    </Methods>
  </Command>
</CommandList>
```

Figure 262 An example of **GetTrunkCllisByGatewayName** response

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <GetTrunkCllisByGatewayName>
        <TrunkClli>PRIFORUPGRADEDOC</TrunkClli>
      </GetTrunkCllisByGatewayName>
    </Methods>
  </Response>
</CommandList>
```

Figure 263 An example of **GetGatewayNames** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <GetGatewayNames
    />
    </Methods>
  </Command>
</CommandList>
```

Figure 264 An example of **GetGatewayNames** response

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <GetGatewayNames>
        <Gateway Names="GAOPRI,PRIFORDOC,PVG1"/>
      </GetGatewayNames>
    </Methods>
  </Response>
</CommandList>
```

Figure 265 An example of **GetCarriers** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <GetCarriers
    />
    </Methods>
  </Command>
</CommandList>
```

Figure 266 An example of **GetCarriers** response

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <GetCarriers>
        <Carriers GatewayName="PVG1" GWCName="GWC-112" CarrierNames="E1_1111,E1_1112"/>
        <Carriers GatewayName="GAOPRI" GWCName="GWC-0" CarrierNames="E1_1001"/>
        <Carriers GatewayName="PRIFORDOC" GWCName="GWC-0" CarrierNames="DS1_10,DS3_10.1"/>
      </GetCarriers>
    </Methods>
  </Response>
</CommandList>
```

The GetCMCLLI command is available till SN08. In SN08 this command is replaced by GetCMCLLibySDMIP

Figure 267 An example of **GetCMCLLI** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <GetCMCLLIs
    />
    </Methods>
  </Command>
</CommandList>
```

Figure 268 An example of **GetCMCLLI** response

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <GetCMCLLIs>
        <CallServer Names="COMPACT5"/>
      </GetCMCLLIs>
    </Methods>
  </Response>
</CommandList>
```

Figure 269 An example of **GetCMCLLibySDMIP** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <GetCMCLLibySDMIP SDMIP="47.140.120.11" />
    </Methods>
  </Command>
</CommandList>
```

Figure 270 An example of **GetCMCLLIbySDMIP** response

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <GetCMCLLIbySDMIP>
        <CMCLLI Names="COMPACT6"/>
      </GetCMCLLIbySDMIP>
    </Methods>
  </Response>
</CommandList>
```

Figure 271 An example of **GetGWCNames** request:

```
<?xml version="1.0"?>
<CommandList>
  <Command>
    <Interface>TrunkMtc</Interface>
    <Methods Version="2.0">
      <GetGWCNames
      />
    </Methods>
  </Command>
</CommandList>
```

Figure 272 An example of **GetGWCNames** response

```
<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <GetGWCNames>
        <GWC Names="GWC-112,GWC-4,GWC-1,GWC-2,GWC-0"/>
      </GetGWCNames>
    </Methods>
  </Response>
</CommandList>
```

Figure 273 An example of error response

```

<?xml version='1.0'?>
<CommandList>
  <Response>
    <Interface>TrunkMtc</Interface>
    <Methods>
      <RTSDChannelByTid>
        <Header
          CMCLLI="COMPACT5"
        />
        <Members>
          <Member>
            <Error
              Number="1311"
              Message="Unable to transition the endpoint to the requested state"
              Severity="MAJOR"
            />
          </Member>
        </Members>
      </RTSDChannelByTid>
    </Methods>
  </Response>
</CommandList>

```

13.5 User Authorization for Trunk Maintenance Operation

In addition to users belonging to “succssn” group to login to OSSGate, user need to be in application specific groups to perform specific operations. Each operation is associated with one or more user groups. In order to execute a command, a user must belong to at least one of the associated user groups. The user groups associated with each TMM maintenance operation are specified in Table-16.

Table 19: TMM Commands Authorization levels

Command	User	Group			
	trkadm	trkrw	trkmtc	trksprov	trkro
Post by Trunk CLLI	X	X	X	X	X
D-Channel Post by Trunk CLLI	X	X	X	X	X
Mtc by Trunk CLLI (BSY, RTS, INB, FRLS)	X	X	X		
ICOT	X	X	X		

Table 19: TMM Commands Authorization levels

Command	User	Group			
	trkadn	trkrw	trkmtc	trksprov	trkro
D-Channel Mtc by Trunk CLLI (BSY, RTS,BSYINB)	X	X	X		
Post by Gateway Name	X	X	X	X	X
QES by Gateway Name	X	X	X	X	X
Mtc By Gateway Name (BSY, RTS, INB, FRLS)	X	X	X		
Post by Carrier	X	X	X	X	X
QES by Carrier	X	X	X	X	X
Mtc By Carrier (BSY, RTS, INB, FRLS)	X	X	X		

13.6 Limitations and Restrictions

- 1 Only individual trunk members (no lists or ranges) may be specified when performing an ICOT test. If an ICOT test is requested on an ISUP trunk which is not in the BSY state, the request will be rejected.
- 2 Maintenance operations are supported for ISUP and PRI trunks. Other trunk types may work, but they are not explicitly supported.
- 3 Post Endpoints by Gateway name is supported.
- 4 Only one maintenance (RTS, BSY, BSY INB or FRLS) request, plus either POST or QES, is allowed per XML request. Batching of maintenance requests is not allowed.
- 5 TMM supports the following commands: BSY, RTS, FRLS, BSY INB, Post (by Trunk CLLI) and QES (by gateway name) for ISUP, PRI and PTS trunks. Other trunk types may work, but they are not explicitly supported.

14: TL1 Line Test Interface for MG9000 Lines

The TL1 Open line test access interface allow test OSS vendors to operate line test access relays. A TL1 interface in the CS2K management tool system provides a common point of access for MG9K line test. TL1 provides an accepted, standards-based interface and command set as an overlay to pre-existing line test functionality on the MG9K. Succession Networks MG9000 line gateway supports POTS, coin, P-phone, ground start, 8x8 POTS and ADSL and DS-1 circuits. Line test access is via MTA (Metallic Test Access) matrix built in the gateway. Each gateway has eight (8) test access ports available; each of which supports 2-wire (one tip-ring pair) test access. Test directions supported by the circuits are:

- test in (towards the circuit)
- test out (towards subscriber loop)
- test bridged configuration.

Although the high impedance isolation is not provided by the MTA in test bridged configurations, it is believed that nearly all currently available test heads have such isolation built in them. The MTA is designed to support both narrowband and wideband line test, provided the configuration meets Nortel Networks' guide lines.

A MG9000 node can have multiple frames which can house up to four (4) shelves each.

Each shelf has multiple cards serving subscribers. Each card has multiple circuits which each support a subscriber loop. In an MG9000 node, any test access port can reach any loop for testing. A maximum of two concurrent line test access connections can be established on each shelf. One concurrent line test connection can be established per card.

14.0.1 Open Line Test Interface Coverage

The following will be supported in SN08:

- Line test access of MG9000 lines.
- Line testing on all Succession Networks supported POTS lines, 8x8 POTS & ADSL lines on MG9000;.
- Both narrowband and wideband line test wherever applicable; although the test access ports only support 2-wire test access

14.0.2 Communication and Interface Protocols

The Test System Controller (TSC) communicates with the CMT (Call server Management Tools) for line test access request. CMT is connected to a secured IP network. The communication between TSC and CMT is via the SSH (Secured SHell). Details on the configuration of SSH and OSSGate usage can be found in the OSSGate User's Guide (NTP# NE10004-512).

Once logged in to the OSSGate, the user will need to change modes to enable the TL1 interface.

From the OSSGate prompt, the user will enter ^B (control-B)

```
>^B <enter>
?
```

A question-mark will be displayed indicating the user is now in control mode. To change to TL1 mode, enter:

```
? mode tl1 <enter>
```

The user will see a message:

```
mode is TL1
>
```

OSSGate is now ready to forward TL1 commands to the TL1 server for processing.

14.0.3 User Authorization

Each TL1 command is assigned to one or more user authorization groups. For an OSSGate login user to have permission to execute a TL1 command, the user must belong to an authorization group associated with the command. The

following table provides a mapping of TL1 commands to the required group membership of the user running them.

TL1 Command	Permitted Authorization Groups
CONN-SLACC-MET	Inmtc, Inadm, Inrw
CHG-SLACC-DIR	Inmtc, Inadm, Inrw
CHG-SLACC-BW	Inmtc, Inadm, Inrw
REPT-STAT	Inmtc, Inadm, Inrw
DISC-SLACC	Inmtc, Inadm, Inrw
RTRV-SLACC-INFO	Inmtc, Inadm, Inrw, Inro

Members of “linemtc, Inadm, and Inrw ” have permission to run all TL1 line test commands. The “Inro” authorization group restricts users to query functions only via the RTRV-SLACC-INFO command.

Users can be assigned to different authorization groups with the “usermod” command from the UNIX command line on the CMT server. Only the root user is permitted to run this command. See document: NN10276-500 “ATM/IP Configuration Management” for details

14.0.4 Line Test Access Functionality

14.0.4.1 TL1 Interface Behavior and Dependencies

- 1 Users are required to login to the OSSGate on the CMT via the SSH protocol using a user ID and password.
- 2 The same user ID and password can be used to login more than one TL1 session.
- 3 The MG9000 maintains an inactivity timeout for line test connections of 180 seconds. If the timeout expires, the TL1 user’s existing test access connection will be disconnected and further commands will fail until a new test connection is established. The user login session will not be impacted by this timeout.
- 4 The OSSGate application maintains an inactivity timeout of 5 minutes (300 seconds). If this timeout expires, the TL1 user will be logged out of the system.

- 5 Each TL1 login session can have multiple active test connections. The maximum number of test connections of each TL1 login user will be determined during the implementation.
- 6 There will not be unsolicited messages sent to the user. One TL1 request can expect one TL1 response.
- 7 Directory numbers **must** be specified in their fully qualified form (FQDN) for commands that operate on test access relays. It is optional *only* for the RTRV-SLACC-INFO.

14.0.5 SN08 : TL1 Line Test Access Command Summary

Command Code	Description
CONN-SLACC-MET	Connect the subscriber loop on the test bus.
CHG-SLACC-DIR	Set the test direction to IN (towards the line circuit), OUT (towards the subscriber loop) or BRIDGED.
CHG-SLACC-BW	Change the test access band between narrowband access and wideband access.
REPT-STAT	Reset the 180 second inactivity timer in the system.
DISC-SLACC	Reset all the test access relays to their default state.
RTRV-SLACC-INFO	Retrieve a subscribers' line information.

Note: Transaction Language 1 (TL1) is the line test access protocol used for this interface. It is based on the Telcordia specifications GR-831-CORE and GR-834-CORE. The following sections describe the commands and syntax supported by this interface.

14.0.6 TL1 Syntax Rules:

The following are the TL1 syntax rules enforced by this implementation:

- All TL1 fields are delimited by “:” characters.
- The first 5 fields are mandatory - even if blank. This includes Command code, Target Id, Access Id, Correlation Tag, and General Block.
- If a specific field allows white-space, then it must be enclosed by double quotes “” if the white-space is to be preserved, otherwise it is ignored.
- Each field may consist of one or more parameters separated by white-space or optional comma. This implementation uses commas to delimit parameters within a given field.

These components will each be discussed in turn.

14.0.7 Command Code - (mandatory)

This is the actual command name to be executed and is typically defined in several parts:

<TL1Verb>-<Modifier1>-<modifier2>

A dash “-” character is used to delimit these components.

For this feature, the following command codes are supported:

- CONN-SLACC-MET
- CHG-SLACC-DIR
- CHG-SLACC-BW
- RTRV-SLACC-INFO
- REPT-STAT
- DISC-SLACC

Valid characters for this field are: letters, digits, or hyphen “-”. Any other characters found in this field will cause the operation to fail. This field is **not** case sensitive.

14.0.7.1 Target Identifier (TID) - (mandatory)

As per the TL1 specification (GR-831), this field can consist of up to 20 characters and must be letters, digits, or hyphens. In this implementation, *it is used to define the address format* of the request and must be set to one of the following values:

- DN
- LEN

Although this field is mandatory, certain commands require that it be left empty. This field is **not** case sensitive.

14.0.7.2 Access Identifier Block - (mandatory)

This field can consist of one or more parameters. The parser mainly checks for illegal white-space in unquoted parameters.

In this implementation, this field is used to specify the DN or LEN to be tested and is **not** case sensitive.

14.0.7.3 Correlation Tag (CTAG) - (mandatory)

This field can have a maximum size of 6 alphanumeric characters. This is a mandatory field for all commands. This field is case sensitive.

14.0.7.4 General Block - (mandatory)

This field is not used in this implementation. However, it is still mandatory and must be specified as an empty field.

14.0.7.5 Payload Block - (optional)

The payload block contains user-defined data and can consist of zero or more fields. Each Field may contain one or more data elements separated by commas.

Parameters within each field can be specified as either *Positional* or *Name-defined*

- **Positional parameters** have a fixed number of components. Empty components are specified by omitting the data, but inserting the delimiter in its place to show the data as missing. The rules are basically the same as that used when using CSV. All positional components must be specified even if the data is not present.
- **Named-defined** parameters specify their components using a NAME=VALUE approach. The order of components is not enforced and can be omitted if desired.

Important Note: For this feature, all commands defined contain either zero or one parameter in the payload block. Only Name-defined parameters are used and contain a variable number of components (depending on the command). All tag names and enumerated values are case insensitive. All other values are treated as case sensitive.

14.0.8 TL1 Responses

There are two types of TL1 response message formats used in this feature: (note that spaces are represented by the ^ (caret) character).

14.0.8.1 Typical response containing header and optional parameter fields**Success Response:**

```
<cr> <lf> <lf>
^^^<af>^<date>^<time> <cr> <lf>
M^^<ctag>^<cs> <cr> <lf>
```



```
^^^["<optional comma-delimited parameters>"]
<cr> <lf>;
```

where:

<af>: address format

<lf>: line feed

<cr>: carriage return

<cs>: command state (COMPLD == success, DENY == error)

Error Message Format:

```
<cr><lf><lf>
^^^<af>^<date>^<time> <cr><lf>
M^^<ctag>^DENY <cr><lf>
^^^<TL1 ERROR CODE><cr><lf>
^^^<optional comma-delimited error parameters><cr><lf>
^^^/*<GR-834 error code description>*/<cr><lf>
^^^/*<System-specific error description>*/<cr><lf>
;
```

Error Message Example:

```
> CONN-SLACC-MET:DN:1418420217:CTAG2::TAP=2,BW=NB,DIR=O,sz=1;
```

```
DN 04-06-17 14:13:51
M CTAG2 DENY
IPMS
DN=1418420217,TYPE="SINGLE PARTY LINE",LEN="UAIP 00 2 02
17",OPTIONS=DGT,GW_NAME=UAIP000-0-2,EP_NAME=tp/02/17,GW_NAME=UAIP000
-0-2,NE_NUM=2,LF=UE9000MG,LS=IDL,CLLI=COMPACT3,BW=NB,DIR=O,sz=1
/* PARAMETER MISSING */
/* Missing command parameter "FA". */
;
```

14.0.8.2 Acknowledgement

This response type is used by the application to construct TL1 acknowledgements to user commands. This type of TL1 response contains no data. It is used to simply ACK specific queries. In this implementation, it is returned as a response to the REPT-STAT command.

```
OK^<CTAG><cr><lf>
<
> rept-stat::8420217:ctag2;
```

```
OK ctag2
<
```

14.0.9 Concurrent TL1 Operations

In SN08, the TL1 Interface for MG9K line test restricts the number of concurrent TL1 commands to 5. Although the number of TL1 login users may exceed this value, the TL1 server will limit the number of TL1 commands that can execute simultaneously on the CS2M. When all resources are busy, a response similar to the following will be returned to indicate that resource exhaustion has (temporarily) occurred.

```
DNLEN 04-10-13 09:51:49
M UNKNWN DENY
ESPG
/* SOFTWARE PROGRAM */
/* Error creating command instance for command "TL1Transaction"
pool.ResourceUnavailableException: The maximum number of concurrent
instances of this command has been reached, please try later. */
;
```

This problem typically occurs when more than 5 TL1 OSS clients are simultaneously logged in and executing commands at the same time. Normally this does not occur with TL1 login users entering commands manually.

To avoid this condition, it is recommended that the number of TL1 user logins does not exceed the concurrent operations threshold (5).

14.0.10 Connect Subscriber Loop on the Test Bus (CONN-SLACC-MET)

The command CONN-SLACC-MET connects the subscriber loop on the test bus. The input, output and the behavior of this command are described in the following sections.

14.0.10.1 Input Fields

- Syntax:

```
CONN-SLACC-MET:<af>:<DN|LEN>:<ctag>::TAP=<tap>,BW=<bw>,DIR=<dir>,FA=
<fa>,SZ=<sz>;
```

Connect a subscriber line on the test bus (CONN-SLACC-MET)	Description
Input:	
AF (Address Format) (Mandatory input)	For DN access, enter DN . For LEN access, enter LEN
DN, LEN (Mandatory input)	
DN format (# of digits)	If DN is entered, only numerical digits (0-9) are valid input. Min # of digits: 1 Max # of digits: 30 In order to support both North American and international DN format, the TL1 command parser will not reject the number of digits within MIN and MAX above. If the number of digits is invalid to the call server, it should be rejected by the call server.
LEN format	This is the traditional Line Equipment Number stored in CS2K Core. If LEN is entered, alphanumeric character values (0-9, A-Z, a-z) are valid input. The TL1 server converts all characters to upper case before processing the command. LEN format in call server will be used, i.e. SITE name + LEN number For example: CARY 000 1 01 12 CARY is the physical location where the line is originating. 000: physical frame number in Cary 1: shelf number 01: slot number 12: circuit number leading zero (0) is required. For Multiple Appearance Directory Number (MADN), line test access is via LEN only. Note: LENs must be enclosed in double quotes ""
CTAG (correlation tag) (Mandatory input)	Correlation tag (max 6 ASCII characters)
TAP (Test Access Port) (0-999) (Mandatory input)	Test access port ID For MG9000, it is 1 to 8.
BW (Access Bandwidth) (Mandatory input)	

Connect a subscriber line on the test bus (CONN-SLACC-MET)	Description
NB	Narrowband. For MG9000 POTS and SAA card, there is only one test access path. NB or WB will result in the test access on the same path, i.e., no difference. For the 8x8 ADSL card on MG9000, NB line test access allows for the narrowband test access. The test access point is between the POTS circuit card and LPF (Low Pass Filter), when the test direction is OUT (towards the equipment side or subscriber loop side), the test head sees the subscriber loop through LPF. This test access does not cut off the ADSL service.
WB	Wideband Currently is for the 8x8 ADSL card on MG9000 wideband access. The test access is at the subscriber loop side of the splitter. It cuts off both POTS and ADSL services. In Phase 1.1, the WB connection option will not be rejected, unless it is an ADSL circuit.
DIR (Initial test DIRection) (Mandatory input)	
O (OUT)	For lookout towards the subscriber loop. It is also called Equipment side.
B (BRIDGED)	Tap the test pair from the test head to the subscriber line. High impedance isolation is not provided.
I (IN)	For look in towards the circuit.
FA (Forced access flag) (Mandatory input)	
Flag=1, if the line state is CPB or CPD, set the connection up and in BRIDGED mode. Do not allow the user to change the test direction. Flag=2, if the line state is CPB or CPD, reject the connection state. See the table below for the detailed description of the behavior of the forced access flag and seize fields.	Flag=1 Flag=2

Connect a subscriber line on the test bus (CONN-SLACC-MET)	Description
SZ (SeiZe) (Mandatory input)	
1	Seize the line This mode allows the test head to perform the loop testing. However, it does not allow the test head to draw dial tone test. The POTS circuit will not supply the dial tone. In this mode, the voice service is in out-of-service state. The call server will route the caller to this line for a treatment, e.g. supply busy tone or an announcement
2	Do not seize the line. This mode allows the test head to perform not only the loop testing, but draw dial tone test when the test direction is in either bridged state or in state. In this mode, the voice service on the line will behave as follows: The line remains in service. The caller to this line will get the normal ring back tone. Since the called party is under test, the loop is cut off. The called party will not hear the ringing tone

- MG9000 Responses to TL1 Force and Seize Parameters

Request Parameters		MG9000 Response				Notes	
		Line is in an Active Call State		Line is Idle			
Force	Direction	Connection	Seized	Connection	Seized		
No	Split-In	Request Rejected (None)	N/A	None	No	Disruptive directions and split-c only allow the line is	
	Split-Out			None			
	Bridged			Bridged			
Yes	Split-In	None	No	None	Yes		
	Split-Out	None		None			
	Bridged	Bridged		Bridged			
No	Split-In	Request Rejected	N/A	Split-In	Yes		
	Split-Out			Split-Out			
	Bridged			Bridged			
Yes	Split-In	Bridged	No	Split-In			
	Split-Out	Bridged		Split-Out			
	Bridged	Bridged		Bridged			

This table is valid for CONNSLACC-MET commands. In cases where the initial connection has been made, this table is also valid for CHG_SLACC_DIR commands.

When the MG9000 seizes a line for testing, the line state at the call server becomes SB.

For the purposes of this table, the call state column headings map to the following call server states:

- Active Call States: CPB, CPD.
- Idle States: IDL, INB, MB, PLO, LO.

Regarding Line States not supported by this feature—DEL, LMB, NEQ, SB³. The TL1 interface implementation in SESM checks for these call server line states. For lines in these states, connection requests are not forwarded to the MG9000.

³ Testing is not supported if SB is the initial state. This does not apply to lines that have gone SB because the MG9000 has seized them for testing.

TL1 parameter mapping to SNMP MTA request entries:

- The SEIZE parameter maps directly.
- FORCE=Yes in TL1 maps to FORCE=Normal in SNMP.
- FORCE=No in TL1 is new. It will map to the new value ForceDont (6) in SNMP.

The following is a description of the line status and behavior during the line testing:

State	Action	No Seize	Seize
CPB	The CS2K/MG9K must give the test system an indication that it has accessed a line that is in an active call. If desired, the test system can test the line in a non-disruptive bridged state.	Must allow bridged connections.	Must allow split-in, split-out and bridge connections.
	The CS2K/MG9K must not allow a disruptive test action on a line involved in an active call.	Must not affect the line state.	Must set the line to an Out of Service state.
CPD	The CS2K/MG9K must give the test system an indication that it has accessed a line that is in an active call. If desired, the test system can test the line in a non-disruptive bridged state.	Must allow split-in, split-out and bridge connections.	Must allow split-in, split-out and bridge connections.
	The CS2K/MG9K must not allow a disruptive test action on a line involved in an active call.	Must not affect the line state.	Must set the line to an Out of Service state.
DEL	The CS2K/MG9K will not support testing a line in the DEL state.		
IDL	The CS2K/MG9K must allow testing a line in the IDL state. This includes testing with a non-disruptive bridged connection and testing with disruptive split connections.	Must allow split-in, split-out and bridge connections.	Must allow split-in, split-out and bridge connections.
		Must not affect the line state.	Must set the line to an Out of Service state.
INB	The CS2K/MG9K must allow testing a line in the INB state. This includes testing with a non-disruptive bridged connection and testing with disruptive split connections.	Must allow split-in, split-out and bridge connections.	Must allow split-in, split-out and bridge connections.
		Must not affect the line state.	Must set the line to an Out of Service state.
LMB	The CS2K/MG9K will not support testing a line in the LMB state.		

State	Action	No Seize	Seize
LO	The CS2K/MG9K must allow testing a line in the LO state. This includes both non-disruptive bridged connections and disruptive split connections.	Must allow split-in, split-out and bridge connections.	Must allow split-in, split-out and bridge connections.
		Must not affect the line state.	Must set the line to an Out of Service state.

MB	The CS2K/MG9K must allow testing a line in the MB state. This includes both non-disruptive bridged connection and disruptive split connections.	Must allow split-in, split-out and bridge connections. Must not affect the line state.	Must allow split-in, split-out and bridge connections. Must set the line to an Out of Service state.
NEQ	The CS2K/MG9K will not support testing a line in the NEQ state.	Must allow split-in, split-out and bridge connections. Must not affect the line state.	Must allow split-in, split-out and bridge connections. Must set the line to an Out of Service state.
PLO	The CS2K/MG9K must allow testing a line in the PLO state. This includes both non-disruptive bridged connections and disruptive split connections.	Must allow split-in, split-out and bridge connections. Must not affect the line state.	Must allow split-in, split-out and bridge connections. Must set the line to an Out of Service state.
SB	The CS2K/MG9K will not support testing a line in the SB state.		

14.0.10.2 Input format and examples

Syntax:

```
CONN-SLACC-MET:<af>:<DN or LEN>:<ctag>::
TAP=<tap>,BW=<bw>,DIR=<dir>,FA=<fa>,SZ=<sz>;
```

- Example 1 (access via DN):

```
CONN-SLACC-MET:DN:4166213000:CTAG1:: TAP=1,BW=NB,DIR=O,FA=1,SZ=2;
```

- Example 2 (access via LEN):

```
CONN-SLACC-MET:LEN:"CARY 000 1 01
12":CTAG1::TAP=1,BW=NB,DIR=O,FA=1,SZ=2;
```

Within a parameter field, when there is more than one field, a comma ‘,’ is placed between the two adjacent fields as separator. There must not be space before and after the comma.

14.0.10.3 Output Fields

Connect a subscriber line on the test bus (CONN-SLACC-MET)	Description
Output:	

CTAG	Correlation tag
CS (Command Status)	COMPLD: command is executed successfully. DENY: command is executed unsuccessfully.
ACTUALDIR (The actual connection direction) (In, Out, Bridged)	MG9000: I (IN), O (OUT), B (BRIDGED) Note: This parameter is populated only if the actual test direction set by the MG9K is different than requested.
NE Number	The NE NUMBER is a numeric digit defined for each MG9000 node. This ID is unique to all MG9000 nodes associated with the same call server. It is limited to 3 digits. It is not enforced to make sure that it is in sequential order. For example, if the NE NUMBER for the first installed MG9000 node is defined as 0, the NE NUMBER of the next installed MG9000 node does not have to be 1, as long as it is not 0 (it has to be unique). This NE NUMBER is displayed with label: NE Number with the MG9000 node name (NE Name) (See the following screen shot) It is required for the subsequent command request, NE Number and Tap are the mandatory input parameters for the commands after the test connection is set up.
GW_NAME (NE name)	Gateway name defined for MG9000. Notes This is the same as the NE name.
CS2000 CLLI	
LS (Line State)	The Line state before the test access request.
LEN (Equipment Number)	Line Equipment Number (Site name and equipment number) e.g. CARY 000 1 01 09
TYPE (Line Type) (POTS, ISDN, P-phone, etc.)	The Line Type as specified by the QDN/QLEN command
LF (Line Frame type)	For MG9000 return UE9000MG. This field is populated with the gateway type defined in the CMT database.
LCC (Line Class Code)	Line class code of the line as defined in the call server
CARDCODE	Line card type as defined in the CS2000 Examples: GWLPOT, RDTLSG, 6X21AC...



The image shows a software window titled "MG9000 Provisioning View". Inside the window, there is a section labeled "MG9000" followed by a sub-section "Attributes". Under "Attributes", there are five labels with corresponding input fields: "NE Number:" with a single-line text box, "NE Name:" with a single-line text box, "NE IP Address/Hostname:" with a single-line text box, "NE Manager IP Address/Hostname:" with a single-line text box, and "NE Provisioning Mode:" with a dropdown menu currently showing "Auto Discover". At the bottom of the window, there are two buttons: "Apply" and "Close".

Warning:

If the test connection is in the bridged or in state, Nortel recommends that the test controller may only execute passive commands, i.e., those which do not require applying voltage, current, or tones to the loop or line card (such as AC/DC voltage measurements) to subscriber test functions. For example, do not perform the 12 metallic loop test function. The reason is that MG9000 POTS card has a hazard voltage protection feature to protect the POTS circuit from foreign voltage. If the voltage applied on the line in BRIDGED (or IN state) is above the hazardous voltage threshold, the subscriber line will be cut off. This may cut off an active call.

14.0.10.4 Output Format and Examples

- Syntax:

```
<cr> <lf> <lf>
^^^<af>^<date>^<time> <cr> <lf>
M^^<ctag>^<cs> <cr> <lf>
^^^"ACTUALDIR=<dir>"
<cr> <lf>;
```

• Example:

```

M DN 03-12-12 03:23:43
  CTAG2 COMPLD
  "ACTUALDIR= B"Syntax:
  • Example:
    DN 03-12-12 03:23:43
M CTAG2 COMPLD;
```

See Tables 1 and 2 for TL1 Error mapping for the TL1 server and MG9K

14.0.10.5 Dependencies

This command must be rejected, if a connection has already been setup for the same session.

14.0.11 Change Test Direction – CHG-SLACC-DIR

Command CHG-SLACC-DIR allows a user to set the test direction to IN, OUT and BRIDGED. IN means connect the tip-ring test pair to the circuit on a line card. OUT means connect the tip-ring test pair to a subscriber loop. BRIDGED means connect the tip-ring test pair to the line. In the BRIDGED state, the high impedance isolation is not provided by the MTA at test bridged configuration for the Nortel product. This command is valid if CONN-SLACC-MET was executed successfully. The input, output and the behavior of this command are described in the following sections.

14.0.11.1 Input Fields

Change test access direction (CHG-SLACC-DIR)	Description
Input	
AF (Address Format) (Mandatory input)	For DN access, enter DN . For LEN access, enter LEN
DN, LEN (Mandatory input)	
DN format (# of digits)	If DN is entered, only numerical digits (0-9) are valid input. Min # of digits: 1 Max # of digits: 30 In order to support both North American and international DN format, the TL1 command parser should not reject the number of digits within MIN and MAX above. If the number of digits are invalid to the call server, it should be rejected by the call server.

LEN format	This is the traditional Line Equipment Number stored in the CS2K Core. If LEN is entered, alphanumeric character values (0-9, A-Z, a-z) are valid input. The TL1 server converts all characters to upper case before processing the command. The LEN format in the call server will be used, i.e. SITE name + LEN number For example: CARY 000 1 01 12 CARY is the physical location where the line originates. 000: physical frame number in Cary 1: shelf number 01: slot number 12: circuit number leading zero (0) is required. For the Multiple Appearance Directory Number (MADN), line test access is via LEN only. Note: LENS must be enclosed in double quotes ""
CTAG (Mandatory input)	Correlation tag (max 6 ASCII characters)
DIR (Test direction) (Mandatory input)	The test direction change will be rejected, if, for example, the line is in the CPB (Call Processing Busy) state, the test direction is in the BRIDGED state. Changing the test direction to OUT or IN will be rejected, since cut-off of the subscribers' conversation is not allowed. If the test access direction change is rejected, the response must not produce an error message. It should response with COMPLD and with no error message. This behavior is consistent with the current no-test trunk interface behavior. And Access Care does not expect an error message, if the desired test access direction cannot be changed.
O (OUT)	Toward the subscriber loop. If the line is busy and the original state is BRIDGED, changing the line test direction to OUT is not allowed.
B (BRIDGED)	Tap the test pair from the test head to the subscriber line. High impedance isolation is not provided.
I (IN)	Toward the line circuit. If the line is busy and the original state is BRIDGED, changing the line test direction to IN is not allowed.

14.0.11.2 Input Format and Examples

- Syntax:

CHG-SLACC-DIR : : <DN> | <LEN> : <ctag> : : DIR=<dir>;

- Example:

CHG-SLACC-DIR::4166213000:CTAG2::DIR=I;

14.0.11.3 Output Fields

Change test access direction (CHG-SLACC-DIR)	Description

Output	
CS (Command Status)	COMPLD: command is executed successfully. DENY: command is executed unsuccessfully.
CTAG	Correlation tag
ACTUALDIR (The actual connection direction)	MG9000: I (IN), O (OUT), B (BRIDGED) Note: This parameter is populated only if the actual test direction set by the MG9K is different than requested.

14.0.11.4 Output Format and Examples

Syntax:

```
<cr> <lf> <lf>
^^^<af>^<date>^<time> <cr> <lf>
M^^<ctag>^<cs> <cr> <lf>
^^^"ACTUALDIR=<dir>"
<cr> <lf>;
```

Examples:

```
DN 03-12-12 03:23:43
M CTAG2 COMPLD
"ACTUALDIR= B"
```

```
DN 03-12-12 03:23:43
M CTAG2 COMPLD
```

Syntax:

```
<cr><lf><lf>
^^^<af>^<date>^<time> <cr><lf>
M^^<ctag>^DENY <cr><lf>
^^^<TL1 ERROR CODE><cr><lf>
^^^[''<optional comma-delimited error parameters>"]<cr><lf>
^^^/*<TL1 error description><cr><lf>*/
^^^/*<System-specific error description>*/<cr><lf>
;
```

See Tables 1 and 2 for TL1 Error mapping for the TL1 server and MG9K

14.0.11.5 Dependencies

This command is valid only

- After the successful execution of CONN-SLACC-MET.
- The line is not BUSY

14.0.12 Change Test Access Bandwidth – CHG-SLACC-BW

Command CHG-SLACC-BW allows for changing the test access between narrowband access and wideband access. In MG9000, the POTS and SAA cards have only one physical test access. It supports wideband testing. CHG-SLACC-BW will not physically change the test access relays. The 8x8 ADSL card has different test access. If the narrowband test access is requested, the test access point will be between the voice circuit and LPF (Low Pass Filter). In this case, the voice service will be cutoff and the ADSL service will still be available to the customer. If the wideband test access is requested, the cutoff relay is on the customer loop side. Both the voice and the ADSL service will be cutoff. The CHG-SLACC-BW command is independent from the current test direction. For example, if the current test direction is narrowband OUT, CHG-SLACC-BW will set the test access to wideband OUT and vise-versa. This logic applies to test direction IN and BRIDGED as well. If the line is voice busy during the initial test access, the test direction will be set to BRIDGED. CHG-SLACC-BW is allowed to switch between wideband BRIDGED and narrowband BRIDGED. However, changing CHG-SLACC-DIR to IN or OUT is not allowed in both cases.

14.0.12.1 Input Fields

Change test access direction (CHG-SLACC-BW)	Description
Input	

AF (Address Format) (Mandatory input)	For DN access, enter DN . For LEN access, enter LEN
DN, LEN (Mandatory input)	
DN format (# of digits)	If DN is entered, only numerical digits (0-9) are valid input. Min # of digits: 1 Max # of digits: 30 In order to support both North American and international DN format, the TL1 command parser should not reject the number of digits within MIN and MAX above. If the number of digits is invalid to the call server, it should be rejected by the call server.
LEN format	This is the traditional Line Equipment Number stored in the CS2K Core. If LEN is entered, alphanumeric character values (0-9, A-Z, a-z) are valid input. The TL1 server converts all characters to upper case before processing the command. The LEN format in the call server will be used, i.e. SITE name + LEN number For example: CARY 000 1 01 12 CARY is the physical location where the line originates. 000: physical frame number in Cary 1: shelf number 01: slot number 12: circuit number leading zero (0) is required. For Multiple Appearance Directory Number (MADN), line test access via LEN only. Note: LENs must be enclosed in double quotes ""
CTAG (Mandatory input)	Correlation tag (max 6 ASCII characters)
BW (Access Bandwidth) (Mandatory input)	
NB	Narrowband. For the MG9000 POTS and SAA card, there is only one test access path. NB or WB will result in the test access on the same path, i.e., no difference. For the 8x8 ADSL card on MG9000, NB line test access allows for the narrowband test access. The test access point is between the POTS circuit card and LPF (Low Pass Filter), when the test direction is OUT (subscriber loop side), the test head sees the subscriber loop through LPF. This test access does not cut off the ADSL service.
WB	Wideband Currently this is for the 8x8 ADSL card on MG9000 wideband access. The test access is at the subscriber loop side of the splitter. It cuts off both the POTS and ADSL services.

14.0.12.2 Input Format and Examples

Syntax:

CHG-SLACC-BW : : <DN> | <LEN> : <ctag> : : BW=<bw>;

Example:

CHG-SLACC-BW::4166213000:CTAG3::BW=NB;

Output Fields

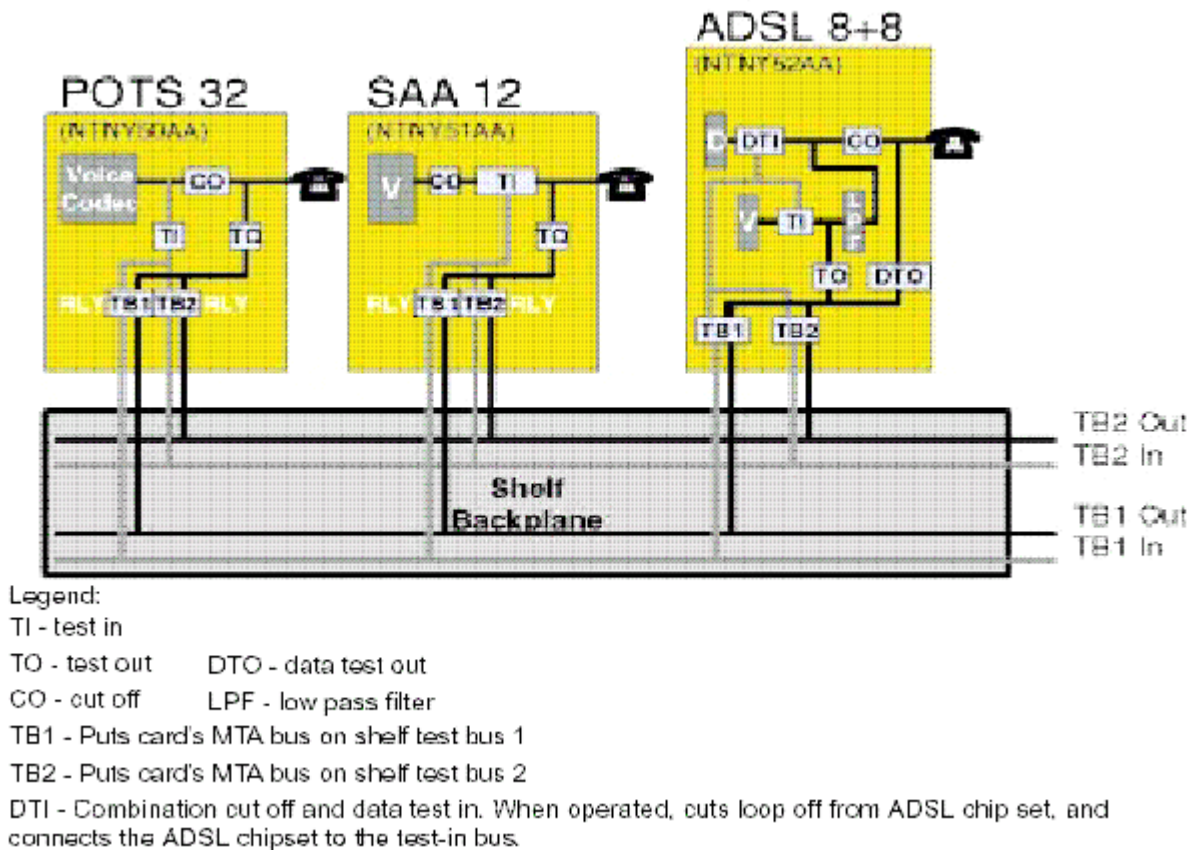
Change test access direction (CHG-SLACC-BW)	Description
Output	
CTAG	Correlation tag
CS (Command Status)	COMPLD: command is executed successfully. DENY: command is executed unsuccessfully.
BW (Access Bandwidth)	N for Narrowband. W for Wideband. Note: For the MG9000 POTS and SAA card, there is only one test access path. NB or WB will result in the test access on the same path, i.e., no difference. For the 8x8 ADSL card on MG9000, NB line test access allows for the narrowband test access. The test access point is between the POTS circuit card and LPF (Low Pass Filter), when the test direction is OUT (towards the equipment side or the subscriber loop side), the test head sees the subscriber loop through LPF. This test access does not cut off the ADSL service. Currently this is for the 8x8 ADSL card on MG9000 wideband access. The test access is at the subscriber loop side of the splitter. It cuts off both POTS and ADSL services.

14.0.12.3 Output Format and Examples**Syntax:**

```
<cr> <lf> <lf>
^^^<af>^<date>^<time> <cr> <lf>
M^^<ctag>^<cs>
<cr> <lf>;
```

Example:

```
DN 03-12-12 03:23:43
M CTAG2 COMPLD
;
```

14.0.12.4 Error Response

Syntax:

```
<cr><lf><lf>
^^<af>^<date>^<time> <cr><lf>
M^^<ctag>^DENY <cr><lf>
^^<TL1 ERROR CODE><cr><lf>
^^["<optional quoted,comma-delimited error parameters>"]<cr><lf>
^^/*<TL1 error description><cr><lf>*/
^^/*<System-specific error description>*/<cr><lf>
```

See Tables 1 & 2 for TL1 Error mapping for the TL1 server and MG9K

14.0.12.5 Dependencies

This command is valid after the successful execution of CONN-SLACC-MET.

14.0.13 Report Status – REPT-STAT

Command REPT-STAT is used for refreshing the 180-second access disconnect timer kept by the gateway. If the gateway does not receive a valid command within the timeout period, it disconnects test access from the subscriber circuit that was specified in the initial CONN-SLACC-MET command.

14.0.13.1 Input Fields

Report status (used for timeout refresh) (REPT-STAT)	Description
Input	
AF (Address Format) (Mandatory input)	For DN access, enter DN . For LEN access, enter LEN
DN, LEN, EPI (Mandatory input)	
DN format (# of digits)	If DN is entered, only numerical digits (0-9) are valid input. Min # of digits: 1 Max # of digits: 30 In order to support both North American and international DN format, the TL1 command parser should not reject the number of digits within MIN and MAX above. If the number of digits is invalid to the call server, it should be rejected by the call server.
LEN format	This is the traditional Line Equipment Number stored in the CS2K Core. If LEN is entered, alphanumeric character values (0-9, A-Z, a-z) are valid input. The TL1 server converts all characters to upper case before processing the command. The LEN format in the call server will be used, i.e. SITE name + LEN number For example: CARY 000 1 01 12 CARY is the physical location where the line originates. 000: physical frame number in Cary 1: shelf number 01: slot number 12: circuit number leading zero (0) is required. For Multiple Appearance Directory Number (MADN), line test access is via LEN only. Note: LENs must be enclosed in double quotes ""
CTAG (Mandatory input)	Correlation tag (max 6 ASCII characters)

14.0.13.2 Input Format and Examples

Syntax:

```
REPT-STAT::<DN>|<LEN>:<ctag>;
```

Example:

```
REPT-STAT::4166213000:CTAG4;
```

14.0.13.3 Output Fields

Report status (used for timeout refresh) (REPT-STAT)	Description
Output	
OK	
CTAG	Correlation tag

14.0.13.4 Output Format and Examples

Syntax:

```
OK <ctag> <cr> <lf>
<
```

Example:

```
OK CTAG2 <cr><lf>
<
```

14.0.13.5 Error Response:

Syntax:

```
<cr><lf><lf>
^^<af>^<date>^<time> <cr><lf>
M^^<ctag>^DENY <cr><lf>
^^<TL1 ERROR CODE><cr><lf>
^^["<optional comma-delimited error parameters>"]<cr><lf>
^^/*<TL1 error description><cr><lf>*/
^^/*<System-specific error description>*/<cr><lf>
```

See Tables 1 and 2 for TL1 Error mapping for the TL1 server and MG9K

14.0.13.6 Dependencies

This command is valid after the successful execution of CONN-SLACC-MET.

14.0.14 Disconnect the Test Connection – DISC-SLACC

The command DISC-SLACC disconnects the test connection. This command restores the circuit under test to its through state and opens all TAP connections to that access point. The input, output and the behavior of this command are described in the following sections.

14.0.14.0.1 Input Fields

Disconnect a subscriber line from the test bus (DISC-SLACC)	Description
Input	
AF (Address Format) (Mandatory input)	DN access, enter DN. For LEN access, enter LEN
DN, LEN (Mandatory input)	
DN format (# of digits)	If DN is entered, only numerical digits (0-9) are valid input. Min # of digits: 1 Max # of digits: 30 In order to support both North American and international DN format, the TL1 command parser should not reject the number of digits within MIN and MAX above. If the number of digits is invalid to the call server, it should be rejected by the call server.
LEN format	This is the traditional Line Equipment Number stored in CS2K Core. If LEN is entered, alphanumeric character values (0-9, A-Z, a-z) are valid input. The TL1 server converts all characters to upper case before processing the command. LEN format in call server will be used, i.e. SITE name + LEN number For example: CARY 000 1 01 12 CARY is the physical location where the line originates. 000: physical frame number in Cary 1: shelf number 01: slot number 12: circuit number leading zero (0) is required. For Multiple Appearance Directory Number (MADN), line test access via LEN only. Note: LENs must be enclosed in double quotes ""
CTAG (Mandatory input)	Correlation tag (max 6 ASCII characters)

14.0.14.1 Input Format and Examples

Syntax:

DISC-SLACC::<DN>|<LEN>:<ctag>;

Example:

DISC-SLACC::4166213000:CTAG4;;

14.0.14.2 Output Fields

Disconnect a subscriber line from the test bus (DISC-SLACC)	Description
Output	
CTAG	Correlation tag
	Phase 1.1
CS (Command Status)	COMPLD: command is executed successfully.
	DENY: command is executed unsuccessfully.
	Phase 1.1

14.0.14.3 Output Format and Examples

Syntax:

```
<cr> <lf> <lf>
^^^<af>^<date>^<time> <cr> <lf>
M^^<ctag>^<cs> <cr> <lf>
<cr> <lf>;
```

Example:

```
DN 03-12-12 03:23:43
M CTAG4 COMPLD
;
```

14.0.14.4 Error Response:

Syntax:

```
<cr><lf><lf>
^^^<af>^<date>^<time> <cr><lf>
M^^<ctag>^DENY <cr><lf>
^^^<TL1 ERROR CODE><cr><lf>
^^^["<optional comma-delimited error parameters>"]<cr><lf>
^^^/*<TL1 error description><cr><lf>
^^^<System-specific error description>*/<cr><lf>
```

See Tables 1 and 2 for TL1 Error mapping for the TL1 server and MG9K

14.0.15 Retrieve Line Information – RTRV-SLACC-INFO

Command RTRV-SLACC-INFO queries the CS2000 to obtain information related to a line. This command does not operate the test access relays, such as the ones in MG9000 MTA and line circuit. It can be executed without a connection up. The input, output and the behavior of this command are described in the following sections.

14.0.15.1 Input Fields

Query line info (RTRV-SLACC-INFO)	Description
Input:	
AF (Address Format) (Mandatory input)	For DN access, enter DN . For LEN access, enter LEN
DN, LEN (Mandatory input)	
DN format (# of digits)	If DN is entered, only numerical digits (0-9) are valid input. Min # of digits: 1 Max # of digits: 30 In order to support both North American and international DN format, the TL1 command parser should not reject the number of digits within MIN and MAX above. If the number of digits are invalid to the call server, it should be rejected by the call server.
LEN format	If LEN is entered, alphanumeric character values (0-9, A-Z, a-z) are valid input. The TL1 server converts all characters to upper case before processing the command. The LEN format in the call server will be used, i.e. SITE name + LEN number For example: CARY 000 1 01 12 CARY is the physical location where the line originates 000: physical frame number in Cary 1: shelf number 01: slot number 12: circuit number leading zero (0) is required. For Multiple Appearance Directory Number (MADN), line test access via LEN only. Note: LENs must be enclosed in double quotes ""
CTAG (correlation tag) (Mandatory input)	Correlation tag (max 6 ASCII characters)

14.0.15.2 Input Format and Examples

Syntax:

```
RTRV-SLACC-INFO:<af>:<DN|LEN>:<ctag>;
```

Example 1 (access via DN):

```
RTRV-SLACC-INFO:DN:4166213000:CTAG6;
```

Example 2 (access via LEN):

```
RTRV-SLACC-INFO:LEN:"CARY 000 1 01 12":CTAG6;
```

14.0.15.3 Output Fields

Query line info (RTRV-SLACC-INFO)	Description
Output:	
CTAG	Correlation tag
CS (Command Status)	COMPLD: command is executed successfully. DENY: command is executed unsuccessfully.
NE Number	The NE Number is a numeric digit defined for each MG9000 node. This ID is unique to all MG9000 nodes associated with the same call server. It is limited to 3 digits. It is not enforced to make sure that it is in sequential order. For example, if the NE Number for the first installed MG9000 node is defined as 0, the NE Number of the next installed MG9000 node does not have to be 1, as long as it is not 0 (it has to be unique). This NE Number is displayed with label: NE Number with the MG9000 node name (NE Name) (See the following screen shot). It is required that for the subsequent command request, NE Number and Tap are the mandatory input parameters for the commands after the test connection is setup.
GW_NAME (NE name)	Gateway name defined for the MG9000. Note: This is the same as the NE name.
CLLI	CS2000 CLLI code
LS (Line State)	Line status before the test access request.
LEN (Equipment Number)	Line Equipment Number (Site name and equipment number) e.g. CARY 000 1 01 09

TYPE (Line Type)	Line Type as specified by the QDN/QLEN command Examples: POYS, ISDN, P-phone...
LF (Line Frame type)	For MG9000 return UE9000MG This field is populated with the gateway type defined in the CMT database.
CARDCODE	Line card type as defined in the CS2000 Examples: GWLPOT, RDTLSG, 6X21AC...
EP_NAME	End Point name on the gateway
OPTIONS	Line options on Call Server

14.0.15.4 Output Format and Examples

Syntax:

```
<cr> <lf> <lf>
^^^<af>^<date>^<time> <cr> <lf>
M^^<ctag>^<cs> <cr> <lf>
^^^DN=<FQDN>,TYPE=<LineType>,LCC=<Line Class Code>,CARDCODE=<card
code>, LEN=<LEN>,GW_NAME=<gatewayName>,EP_NAME=<endpointName>,
OPTIONS=<options>,CLLI=<CS2000CLLI>,CLLI=<CS2K CLLI>,NE_NUM=<ne
number>,LF=<frame Type>,NE_NAME=<ne name>,LS=<line state><cr> <lf>;
```

14.0.15.5 Output Example:

```
DN 04-06-30 20:25:40
M CTAG5 COMPLD
"DN=1418400207,TYPE="SINGLE PARTY
LINE",LCC=IBN,CARDCODE=GWLPOT,LEN="UAIP 00 0 02
07",GW_NAME=UAIP000-0-0, EP_NAME=tp/02/07, OPTIONS=DGT,
CLLI=COMPACT3, NE_NUM=1,LF=UE9000MG,LS=IDL"
;
```

14.0.15.6 Error Response:

Syntax:

```
<cr><lf><lf>
^^^<af>^<date>^<time> <cr><lf>
M^^<ctag>^DENY <cr><lf>
^^^<TL1 ERROR CODE><cr><lf>
^^^["<optional comma-delimited error parameters>"]<cr><lf>
^^^/*<TL1 error description><cr><lf>*/
^^^/*<System-specific error description>*/<cr><lf>
```


See Tables 1 and 2 for TL1 Error mapping for the TL1 server and MG9K

14.0.15.7 Dependencies

This command has no dependencies and can be executed at any time. The input DN or LEN can be the same as the DN or LEN of the active connection, but it is not required

14.0.15.8 Error code listings

The following tables list the TL1 error codes and descriptions generated by the TL1 Server and the MG9K.

Table 4 TL1 Error Codes - TL1 Server

TL1 Error Code	TL1 Category	TL1 Error Text	Description
ESPG	EQUIPAGE	SOFTWARE PROGRAM	An error occurred in the application software while processing a request. All accumulated line information up to the point of error will be displayed. A comment line will be displayed in the TL1 response message indicating the source of failure.
ICNV	INPUT	COMMAND NOT VALID	The command code entered is invalid for this interface.
IDNV	INPUT	DATA NOT VALID	One of the input fields contains invalid data. A comment line will be displayed in the TL1 response message indicating the source of failure.
IIAC	INPUT	INVALID ACCESS IDENTIFIER (AID)	An invalid DN or LEN has been found in the access identifier field of the request message. A comment line will be displayed in the TL1 response message indicating the source of failure.

IICM	INPUT	INVALID COMMAND	The command entered is missing required fields. A comment line will be displayed in the TL1 response message indicating the source of failure.
IICT	INPUT	INVALID CORRELATION TAG	The correlation tag entered violates the one of the following constraints: -must be non-null -must be alpha-numeric -must be between 1-6 characters in length A comment line will be displayed in the TL1 response message indicating the source of failure.
IISP	INPUT	INVALID SYNTAX OR PUNCTUATION	Missing or invalid command terminator.
IITA	INPUT	INVALID TARGET IDENTIFIER	An invalid address format has been found in the target identifier field of the request message. A comment line will be displayed in the TL1 response message indicating the source of failure.
IPMS	INPUT	PARAMETER, MISSING	A required parameter has not been specified in the command entered. A comment line will be displayed in the TL1 response message indicating the source of failure.
IPNC	INPUT	PARAMETER NOT CONSISTENT	An entered parameter has both Positional and Named components. As per GR-831 syntax rules, a parameter components must be of the same type. A comment line will be displayed in the TL1 response message indicating the source of failure.
IPNV	INPUT	PARAMETER NOT VALID	An illegal parameter tag or value has been found in the command string. A comment line will be displayed in the TL1 response message indicating the source of failure

ISPC	INPUT	SYNTAX PUNCTUATION	A syntax error has been detected in the command string. An example would be a missing "around a quoted string. A comment line will be displayed in the TL1 response message indicating the source of failure
ITSN	INPUT	INVALID/INACTIVE TEST SESSION NUMBER	The test session for the DN or LEN entered does not exist. Either there has been no test session established yet, or the test session has been terminated due to an inactivity time-out. A comment line will be displayed in the TL1 response message indicating the source of failure
SCSN	STATUS	INVALID COMMAND SEQUENCE	A TL1 command has been entered for which a required requisite command has not yet been executed. For example, CHG-SLACC-DIR cannot be executed successfully until CONN-SLACC-MET has been run to establish a test session. A comment line will be displayed in the TL1 response message indicating the source of failure
SDBE	STATUS	INTERNAL DATA BASE ERROR	An internal database error has been detected during command execution. A comment line will be displayed in the TL1 response message indicating the source of failure
SNVS	STATUS	NOT IN VALID STATE	The target circuit is not in a state valid for line test. A comment line will be displayed in the TL1 response message indicating the source of failure
SROF	STATUS	REQUESTED OPERATION FAILED	This is a general error indicating that the request could not be processed successfully. A comment line will be displayed in the TL1 response message indicating the source of failure.

SRQN	STATUS	INVALID REQUEST	The operation cannot be performed on the specified target. Possible reasons are: -Unsupported gateway type -NLT option on line (no line test) -line is specified by DN and is member of a MADN group. A comment line will be displayed in the TL1 response message indicating the source of failure

TL1 Error Code	TL1 Error Text	MG9K Error Text (user-visible)	MG9K Error Description
RCBY	CIRCUIT BUSY	"Line Busy, Try Again Later"	The MG sends this if the line appears to be in a call.
SNVS	NOT IN VALID STATE	"Line can't be accessed. Check Line at Gateway Manager."	The MG sends this if the linecard has been pulled, the line circuit is busy, or otherwise inaccessible.
ENEQ	NOT EQUIPPED	"Gateway Not Equipped for This Request. Check Gateway Manager"	The MG sends this when the required equipment is missing. Currently for MTA access, this may never be returned by the MG9000. It might be a good return code if the EM finds that there is not a corresponding tap provisioned for the request.
RANB	ACCESS NETWORK BUSY	"A Gateway Test Resource is Busy. Try Again Later."	This is returned when internal resources are all in use, such as verticals and shelf test buses, or if there is another test on the same linecard (there is only one test bus per line card)

SROF	REQUESTED OPERATION FAILED	“Problem with a Gateway Test Resource. Check Gateway Manager”	The MG sends this when there is nothing wrong with the request, but there is something wrong with required equipment on the MG9000. Could not find a better match in the list.
EFON	FEATURE OPTION NOT PROVIDED	“The request or one of its requested options is not supported.”	For example, request for full-split would return not_supported.
SRQN	INVALID REQUEST	“Invalid Request. Check MTAPT and line exists, and for conflicting parameters.”	MG sends this for scenarios such as conflicting parameters, e.g., a test-out connection requested when seize is false (test-out interferes with call-p, so the line must be seized for test-out).
SABT	ABORTED	“Test session abandoned, connection dropped. Check Test System.”	Line test session inactivity time-out expired on the gateway.
SABT	ABORTED	“The gateway encountered a problem. Try Again. If problem recurs, check line and test Resources at Gateway Manager.”	Line test session terminated by the gateway due to a maintenance event or failure.
SROF	REQUESTED OPERATION FAILED	“Try Again. If problem recurs, check line and test Resources at Gateway Manager.”	Miscellaneous internal system error.
SROF	REQUESTED OPERATION FAILED	“Try Again. If problem recurs, check line and test Resources at Gateway Manager.”	Miscellaneous internal system error.
SROF	REQUESTED OPERATION FAILED	“Try Again. If problem recurs, check line and test Resources at Gateway Manager.”	Miscellaneous internal system error.

14.1 Limitations and restrictions

As stated earlier, directory numbers must be specified in their fully qualified form (FQDN) for test commands that

manipulate test connections on the MG9K. This includes the following commands:

- CONN-SLACC-MET
- CHG-SLACC-DIR
- CHG-SLACC-BW
- REPT-STAT

Since RTRV-SLACC-INFO has no dependencies and only queries the call server, FQDN is optional for this command.

14.2 Upgrade Dependencies

This feature introduces a dependency in the upgrade sequence of the Access Care system with the rest of the Succession components.

When upgrading from releases prior to SN07 to SN07 and greater, it is required that Access Care be upgraded **ONLY** after the CMT, MG9K-EM and MG9K upgrades are complete. Failing to do so will disable Line test capabilities between Access care and the MG9K.

15: Glossary

15.1 Terminology & Description of Method Parameters

- **gwcName** - the external or public name to a GWC (Gateway Controller), e.g. GWC7 or CallServerName, GWC7. TYPE String.
- **mgName** - the external name of the MG (Media Gateway), e.g. PVG03, PVG11. TYPE String.
- **RC** - the return code, a numeric value indicating the result of the attempted operation. TYPE String.
- **MsgText** - the appropriate text string that describes the result of the operation.
- **<Component>-ErrorCode** - the error code supplied by the component which generated the error. (e.g.: <Oracle-RC>17002</Oracle-RC>)
- **<Component>-ErrorText** - the text string supplied by the component which generated the error.
- **TN** - terminal number. The valid range is 1-n, where n = 24 for North American carriers (DS1) and n = 31 for International carriers (E1).
- **TID** - Terminal Identifier - consisting of node number (1-4094) and terminal number (1-4094). This numeric identifier is allocated (one per endpoint) and managed by the GWC EM. The GWC EM also maintains a mapping of endpoints to TIDs. TYPE Integer
- **carrierName** - Name of the carrier, following the appropriate endpoint naming convention for the chosen protocol. TYPE String.
- **endPointName** - name of this endpoint; defined using naming convention for the associated protocol. TYPE String.

16: Definitions & Abbreviations

Acronym	Definition
ADSL	Asymmetric Digital Subscriber Line
AFC	Advanced Fibre Communications
BCM	Business Communications Manager
CS	Communication Server
CS2K	Call Server 2000
CORBA	Common Object Request Broker Architecture
DDMS	DMS Data Management System
DMS	Digital Multiplex System
DQoS	Dynamic quality of service
DS	Differentiated services field. An 8 bit field containing the 6 bit DSCP subfield.
DSCP	Diffserv Code Point. A 6 bit subfield of the DS field in every IP packet which identifies the Diffserv Per Hop Behavior. In IP version 4, the TOS byte is redefined to be the DSCP. In IP version 6, the Traffic Class octet is used as the DSCP.
DTD	Document Type Definition
DTC	Digital Trunk Controller
DTCI	ISDN Digital Trunk Controller
EM	Element Manager
EPID	End Point Identifier
GUI	Graphical User Interface
GW	Gateway
GWC	Gateway Controller
GWC EM	Gateway Controller Element Manager
HTML	HyperText Markup Language

IDL	Interface Definition Language
IP	Internet Protocol
LBL	Low Bandwidth Link
LTC	Line/Trunk Controller
MG9K	Media Gateway 9000
MG9K EM	Media Gateway 9000 Element Manager
MG	Media Gateway
MGC	Media Gateway Controller
MP	Media Proxy
MTA	Multimedia terminal adapter
MTC	Maintenance
NAT	Network Address Translator
NV	Network View
NN	Node Number
OSS	Operation Support System
OSSDI	OSS Data Interface
PEP	Policy Enforcement Point
PRI	Primary Rate Interface
RU	Resource Usage
SNMP	Simple Network Management Protocol
SSPFS	Succession Solaris Platform Foundation Software
SS7	Signaling System 7
TDM	Time Division Multiplexing
TID	Terminal Identifier
TMM	Trunk Maintenance Manager
TN	Terminal Number
USN	Unique Sequence Number

XML	Extensible Markup Language
XMLCP	XML Command Processor
XSD	XML Schema Definition
XSL	Extensible Style Sheet Language

Term	Explanation
CMT	CS2000 Management Tools
CPB	Call Processing Busy
CS2000	Call Server 2000
GW	GateWay
LPF	Low Pass Filter
MG9000	Media Gateway 9000 (line gateway)
MTA	Metallic Test Access
MTAP	Metallic Test Access Port on MG9000
MTM	Metallic Test access Module
NE	Network Element
RMM	Remote Maintenance Module
RTU	Remote Test Unit
SSH	Secure Shell
TL1	Transaction Language 1
TSC	Test System Controller
UPC	User Privilege Code

17: SSH Products Comparison

OSSGate supports SSH to enable secure data transport. The table below provides a comparison of several SSH products, which are available in the market. This table is provides as a guidance for selection if needed.

Table 20: SSH Product Comparison

Name	Platform	License	Protocol	Server	Authentication	KeyGen	Contact	Forwarding
AmigaSSH	Amiga	GNU PublicLicense	SSH-1	No	Password, Public-key	ssh-keygen	www.lysator.liu.se/~lilja/amigassh	No
SSH	BeOS	Freeware	SSH-1	No	Password, Public-key, trusted-Host	ssh-keygen	www.bebits.com/app/703	Port, X
JavaSSH	Java	Freely Distributable	SSH-1	No	Password, Public-key	N/A	www.cl.cam.ac.uk/~fapp2/software/java-ssh	No
MindTerm	Java	GNU publicLicense	SSH-1	No	Password, Public-key, Trusted-host, TIS, sid-token	Yes	www.mindbright.se	Port, X
F-Secure SSH client	Mac, Windows	Commercial	SSH-1, SSH-2	No	Password, Public-key	Yes	www.f-secure.com	Port, X
SSHDOS	MS-DOS	GNU publicLicense	SSH-1	No	Password	No	www.vein.hu/~nagyd	No
SSHOS2	OS/2	Info Not Available	SSH-1	No	Password, Public-key, Trusted-host	Yes	ftp://ftp.cs.hut.fi/pub/ssh/old/os2	Port, X
TopGun SSH	PalmOS	Free	SSH-1	No	Password	No	www.isac.cs.berkeley.edu/pilot	No
Ossh	Unix	BSD License	SSH-1	sshd	Password, Public-key, Trusted-host	ssh-keygen	ftp://ftp.nada.kth.se/pub/krypto/ossh	Port, X

Table 20: SSH Product Comparison

Name	Platform	License	Protocol	Server	Authentication	KeyGen	Contact	Forwarding
FISH	VMS	Freeware	SSH-1	No	Password, Public-key, Trusted-host	Yes	www.free.lps.e/fish	No
sshexec.com	VMS	Freeware	SSH-1	Yes	Password, public-key	Yes	www.er6.eng.ohio-state.edu/~jonesd/ssh	
AppGate	Windows, Unix, Mac	Commercial					www.appgate.com	
Metro StateSSH	Windows	GNU public license	SSH-1	No	Password	No	http://csi.mscedu/MSSH	Port
Okhapkin Port	Windows	Free for non commercial use	SSH-1, SSH-2	Sshd	Password, public-key, trusted-host	ssh-keygen	http://miracle.geol.msu.ru/sos	Port, X
PenguinNet	Windows	Shareware	SSH-1	No	Password, public-key, rhosts	Yes	www.siliconcirceus.com	No
SSH Secure Shell	Windows	Free for non commercial use	SSH-2	No	Password, public-key	Yes	www.ssh.com	Port, X
SecureCRT	Windows	Commercial	SSH-1, SSH-2	No	Password, public-key, TIS	RSA, DSA	ww.vandyke.com	Port, X
SecureFX	Windows	Commercial	SSH-2	No	Password, public-key	Yes	www.vandyke.com	No
SecureKoalaTerm	Windows	Shareware	SSH-1, SSH-2	No	Password, public-key	Yes	www.midasoft.com	No
TTSSH	Windows	Freely distributable	SSH-1	No	Password, public-key, trusted-host, TIS	No	www.zip.com.au/~roca/ttssh.html	Port, X
Zoc	Windows	Commercial	SSH-1	No	Password	No	www.emtec.com/zoc	No
SshCE	Windows	Freeware	SSH-1	No	Password	No	www.movsoftware.com/sshce.htm	No

18: Provisioning Conventions for GWC, Media Gateways, VMG, and Endpoints (SN07)

18.1 PVG gateways

Capacities of MGs are defined in the profiles available for selection when the MG is associated to the GWC. Correct selection of capacity is important to get the maximum utilization of your GWC. For example, a trunking GWC supports a maximum of 4094 datafilled ports, however, some PVG VSP configurations support a maximum of 1120 ports. Therefore to get maximum utilization of the GWC, one would provision 3 1120 port VSPs and 1 624 port VSPs. Changing capacity is supported. To get better utilization of the GWC, one could change reserved ports for a GWC to be lower than what is defined in the profile.

In SN05 and later releases, PVGs are also used to host anchor packet resources. We call this an Anchor Packet Gateway or APG. APGs can only be associated with GWCs that are APG capable.

Table 21: PVG MG Profile Values and Configuration Parameters for Provisioning

Profile Name	Service Type	Reserved Port Capacity	Protocol	OSSGATE Protocol	Protocol Version
PVG15K_ASPEN	trunk	1120	ASPEN,	2	2.1
PVG15K_MEGACO	trunk	1120	MEGACO	4	2.1
PVG15K_1000_ASPEN	trunk	1000	ASPEN	2	2.1
PVG15K_1000_MEGACO	trunk	1000	MEGACO	4	2.1
PVG15K_PARTIAL_ASPEN	trunk	624	ASPEN,	2	2.1
PVG15K_PARTIAL_MEGACO	trunk	624	MEGACO	4	2.1
PVG7K_ASPEN	trunk	1008	ASPEN	2	2.1
PVG7K_MEGACO	trunk	1008	MEGACO	4	2.1
PVG_VSP3_ASPEN	trunk	2016	ASPEN	2	2.1
PVG_VSP3_MEGACO	trunk	2016	MEGACO	4	2.1
PVG_APG_ASPEN	APG	1120	ASPEN	2	2.1

Table 21: PVG MG Profile Values and Configuration Parameters for Provisioning

Profile Name	Service Type	Reserved Port Capacity	Protocol	OSSGATE Protocol	Protocol Version
PVG_APG_MEGACO	APG	1120	MEGACO	4	2.1
PVG_APG_VSP3_ASPEN	APG	2016	ASPEN	2	2.1
PVG_APG_VSP3_MEGACO	APG	2016	MEGACO	4	2.1

Table 22: PVG Internally Used MG Profile Values and Configuration Parameters

Profile Name	MG Category	GWC Connectivity Profile Number	Protocol	Protocol Version (internal storage)
PVG15K_ASPEN	large	46	2	2.1
PVG15K_MEGACO	large	58	4	1.0
PVG15K_1000_ASPEN	large	46	2	2.1
PVG15K_1000_MEGACO	large	58	4	1.0
PVG15K_PARTIAL_ASPEN	large	46	2	2.1
PVG15K_PARTIAL_MEGACO	large	58	4	1.0
PVG7K_ASPEN	large	46	2	2.1
PVG7K_MEGACO	large	58	4	1.0
PVG_VSP3_ASPEN	large	46	2	2.1
PVG_VSP3_MEGACO	large	58	4	1.0
PVG_APG_ASPEN	APG	46	2	2.1
PVG_APG_MEGACO	APG	58	4	1.0
PVG_APG_VSP3_ASPEN	APG	46	2	2.1
PVG_APG_VSP3_MEGACO	APG	58	4	1.0

Table 23: GWC Profiles used by PVG Gateways

Profile Name	Tone Data	[[Exec, Term]...]	[[Type, Capacity],...]
TRUNKNA	NORTHAM	[('DTCEX','PRAB'), ('DTCEX', 'ABTRK')]	[(trunks(2),4094), (largeGWs(8), 24)]
TRUNK250	NORTHAM	[('UTR250','PRAB'), ('DTCEX', 'ABTRK')]	[(trunks(2),4094), (largeGWs(8), 24)]
TRUNKINTL	UKADSI	[('DTCEX','PRAB'), ('DTCEX', 'ABTRK')]	[(trunks(2),4094), (largeGWs(8), 24)]
V52TRUNK	UKADSI	[('POTSEX','POTS'), ('DTCEX', 'ABTRK')]	[(trunks(2),3968), (largeGWs(8), 24), (v52(17), 0)]

Table 24: GWC Profiles used by PVG_APG Gateways

Profile Name	Tone Data	[[Exec, Term]...]	[[Type,Capacity],...]
APG_WITH_RA	NORTHAM	[('DTCEX', 'ABTRK')]	[(apgp(4),6048), (apgGWs(10), 3) (ra(13),0)]
APG	NORTHAM	[('DTCEX', 'ABTRK')]	[(apg(4),6048), (apgGWs(10), 3)]

18.1.1 Media Gateway Name

Up to 8 upper case characters starting with an alphabetical character, followed by alphas and numbers

suggested format:

PVG<n...n><lp> where

<nn> is device number of the PVG, which value should be any character in the range 0-9 or a-z or A-Z, or with '-' or '_' embedded in it.

<lp> is the number of the slot of the logical processor card for the VSP in the PVG (recommended slots 6-11)

e.g. PVG046

18.2 H.323 gateways

The maximum port or endpoint capacity for H.323 gateways varies depending on the vendor gateway type. Consult

Table 25: H.323 MG Profile Values and Configuration Parameters for Provisioning

Profile Name	Service Type	Protocol	OSSGATE Protocol	Protocol Version
NORTEL_BCM	H.323, ITRANS	H.323	6	4.0
SUCCESSION_1000	H.323, ITRANS	H.323	6	4.0
CISCO_2600	H.323, ITRANS	H.323	6	4.0
CISCO_3600	H.323, ITRANS	H.323	6	4.0
CISCO_AS5300	H.323, ITRANS	H.323	6	4.0
WESTELL	H.323, ITRANS	H.323	6	4.0
H.323_PROXY	H.323, ITRANS	H.323	6	4.0

vendor documentation for details.

Table 26: H.323 Internally Used MG Profile Values and Configuration Parameters

Profile Name	MG Category	GWC Connectivity Profile Number	Protocol	Protocol Version (internal storage)
NORTEL_BCM	Large	61	H.323	4.0
SUCCESSION_1000	Large	62	H.323	4.0
CISCO_2600	Large	63	H.323	4.0
CISCO_3600	Large	64	H.323	4.0
CISCO_AS5300	Large	65	H.323	4.0
WESTELL	Large	66	H.323	4.0

Table 27: GWC Profiles used by H.323 Gateways

Profile Name	Tone Data	[[Exec, Term]...]	[[Type, Capacity],...]
H.323_NA	NORTHAM	['DTCEX', 'PRAB'] ['DTCEX', 'ABTRK']	[(h323(21),1032), (largegws(8), 200)]
H.323_INTL	UKADSI	['DTCEX','PRAB'], ('DTCEX', 'ABTRK')]	[(h323(21),1024), (largeGWs(8), 200)]

18.3 CVX gateways

Table 28: CVX MG Profile Values and Configuration Parameters for Provisioning

Profile Name	Service Type	Reserved Port Capacity	Protocol	OSSGATE Protocol	Protocol Version
CVX1800_2688	Trunk	2688	DSMCC	3	5.2
CVX600_612	Trunk	612	DSMCC	3	5.2

Table 29: CVX Internally Used MG Profile Values and Configuration Parameters

Profile Name	MG Category	GWC Connectivity Profile Number	Protocol	Protocol Version (internal storage)
CVX1800_2688	Large	47	3	5.2
CVX600_612	Large	47	3	5.2

18.4 Other trunk gateways and Audio Servers

Table 30: UAS, Profile Values and Configuration Parameters for Provisioning

Profile Name	Service Type	Reserved Port Capacity	GUI Protocol	OSSGATE Protocol	Protocol Version
UAS	audio	0 (undefined)	MEGACO	4	1.0
AUDIOCODES	trunk	280	MEGACO	4	1.0
NEURA_GX_ASPEN	trunk	2108	ASPEN,	2	2.1
NEURA_GX_MEGACO	trunk	2108	MEGACO	4	1.0
TGCP	trunk	-	TGCP	6	

Table 31: Internally Used UAS Profile Values and Configuration Parameters

Profile Name	MG Category	GWC Connectivity Profile Number	Protocol	Protocol Version (internal storage)
UAS	audio	48	4	1.0
AUDIOCODES	large	54	4	1.0
NEURA_GX_ASPEN	large	59	2	2.1
NEURA_GX_MEGACO	large	59	4	1.0
TGCP	large	60	6	

Table 32: GWC Profiles used by UAS Gateways

Profile Name	Tone Data	[[Exec, Term]...]	[[Type,Capacity],...]
AUDCNTL	NORTHAM	[['DTCEX', 'ABTRK']]	[(audio(3),4096), (audioGWs(9),16), (bct(14),0), (ipsec(15),0) (conferences(18),0) , (announcements(19),0)]
AUDCNTLINTL	UKADSI	[['DTCEX', 'ABTRK']]	[(audio(3),4096), (audioGWs(9),16), (bct(14),0),(ipsec(15),0) (conferences(18),0), (announcements(19),0)]

18.4.1 Media Gateway Name

A domain name of the Media Gateway in the form of an absolute domain name including the hostname of the device and suitable for lookup using Directory Name Service (DNS). The name must no longer than 32 characters. An example would be:

uas1.ral5.vendor.net

18.4.2 Endpoint Names

Endpoints are not specified during provisioning of Universal Audio Servers

18.5 Other Gateway Controller Profiles

Table 33: Gateway Controller Profile Updates

Profile Name	Tone Data	Service Type	Service Type, Capacity
BICC	NORTAM	DTCEX, ABTRK	dpt(5), 4096 sipt(11), 0
AUDCNTL_RMGC	NORTHAM	DTCEX, ABTRK	audio (3), 4096; audioGWs(9), 16 bct (14), 0; ipsec (15), 0 kerberos (16), 0 conferences (18), 0 announcements (19), 0 rmgcGWs (20),115,000
AUDCNTL_RMGCINTL	UKADSI	DTCEX, ABTRK	audio (3), 4096; audioGWs(9), 16 bct (14), 0; ipsec (15), 0 kerberos (16), 0 conferences (18), 0 announcements (19), 0 rmgcGWs (20),115,000

18.6 Carrier Name and Carrier Endpoint Names

Carrier Endpoints are provisioned in groups representing E1 and DS1 levels. Provisioning at this level gives access to all timeslot for

service provisioning. Services supported in since SN04 include: ISUP trunking, PRI trunking, V5.2 service and APG service for bearer channel anchoring.

Carrier names are case sensitive on the PVG and should be entered in upper case. Descriptions below include both endpoint groups and individual endpoint descriptions. Services can be applied at the DS1/E1 timeslot level.

In addition to the traditional trunking carrier groups, special H.323 carrier groups are supported using carrier operations.

18.6.0.1 E1

E1_<h1><h2>.<g> where

<h1> is the LP (logical processor) number (or slot) of the E1: 1-15 (no padding)

<h2> is the E1 (two digit) number: 01-32

<g> is the channel number: 1-31 (no leading 0)

e.g. Carrier name: E1_220; Carrier Endpoint name (timeslot) 1

18.6.0.2 DS3

DS3_<b1><b2>.<c>.<d> where

<b1> is the LP (logical processor) number (or slot) of the DS3 (recommended slots 2-5): 2-5

<b2> is the DS3 (single digit) port number: 0-1

<c> is the DS1 number within the DS3: 1-28

<d> is the channel in the DS1: 1-24

e.g. Carrier name: DS3_20.3 - Carrier Endpoint name (timeslot) DS3_20.3.1

18.6.0.3 STM-1

SM_<lp><pp><r>_412_<jj><k><l><m>.<ee> (when using ASPEN protocol)

STM/<lp>/<pp>/<r>/VC4VC12/<jj>/<k>/<l>/<m>/<ee>
(When using H.248/Megaco Protocol)

where

<lp> is the LP (logical processor) number (or slot) of the STM-1 interface: 1-15 (no padding) (recommended slots 2-5)

<pp> is the (two digit) port number: 00-03

<r>: is the (one digit) STM level ID, STM-1:1, STM-4:2 (Only STM-1 is supported now, hence valid value is 1)

"_412_" identifies the muxing type VC4VC12.

<jj>: is the (two digit) AUG number within the STM-n (Valid value 01 only, since STM-1 only is supported now)

<k> is the (one digit) TUG-3 number within a VC4: 1-3

<l> is the (one digit) TUG-2 number within a TUG-3: 1-7

<m> is the (one digit) TU number within a TU: 1-3

<ee> 1 or 2 digit number, with VS-12 channel/timeslot: 1-31 (no leading 0)

e.g. carrier name: SM_2011_412_01361

endpoint name: SM_2011_412_01361.1

e.g. STM/2/1/1/VC4VC12/1/3/6/1

18.6.0.4 OC-n/STS-n

SS_<lp><pp><r>_VT15_<jj><l><m>.<e> when using ASPEN protocol

STS/<lp>/<pp>/<r>/VT15/<jj>/<l>/<m>/<e> when using H.248/Megaco protocol

where

<lp> is the LP (logical processor) number (or slot) of the OC-3 interface: 1-15 (no padding)(recommended slots 2-5)

<pp> is the (two digit) port number: 00-03

<r>: 1 digit STS level ID, STS-3:3, STS-12:4 etc (valid value is 3 since only OC3 is supported as of now)

"_VT15_" identifies the muxing type STS-1/VT15.

<jj> is the (two digit) STS-1 number within the STS-n: (valid range is 01-03, since only OC3 is supported as of now)

<l> is the (one digit) VT group number within STS-1: 1-7

<m> is the (one digit) VT number within a VT: 1-4

<e> is the VT1.5 channel/timeslot: 1-24 (no leading 0)

e.g. carrier name: SS_2023_VT15_0161

endpoint name: SS_2023_VT15_0161.1

e.g STS/2/1/3/VT15/1/6/1

18.7 MG9000 Line Gateways

In SN04 as in SN03 capacities of MGs are defined in the profiles available for selection when the MG is associated to the GWC. Therefore, selection of capacity is important to get the maximum utilization of your GWC.

For example, a line GWC supports a maximum of 6400 datafilled ports, however, MG9000 VMGs support 512 circuits. Changing this capacity is not supported in before SN04. A future release will allow lowering the reserved capacity for existing VMGs and to allow the customer to specify reserved port capacity upon entering the association.

Table 34: MG9000 Profile Values and Configuration Parameters for Provisioning

Profile Name	Service Type	Reserved Port Capacity	GUI Protocol	OSSGATE Protocol	Protocol Version
UE9000MG	line	512	MEGACO	5	1.0

Table 34: MG9000 Profile Values and Configuration Parameters for Provisioning

Profile Name	Service Type	Reserved Port Capacity	GUI Protocol	OSSGATE Protocol	Protocol Version
UE9000MG_IP (was: MG9K_IP)	large	512	MEGACO	1.5 0	1.0

Table 35: MK9K Internally Used MG Profile Values and Configuration Parameters

Profile Name	MG Category	GWC Profile Number	Protocol	Protocol Version (internal storage)
UE9000MG	large	50	5	1.0
UE9000MG_IP (was: MG9K_IP)	large	55	5	1.0

Table 36: GWC Profiles used by MG9000 Gateways

Profile Name	Tone Data	[(Exec, Term)...]	[(Type,Capacity),...]
LARGE_LINENA	NORTHAA	[('POTSEX','POTS'), ('KSETEX','KEYSET')]	[(lines(1),6400), (largeGWs(8),27)]
LARGE_LINEINTL	UKADSI	[('POTSEX','POTS'), ('KSETEX','KEYSET')]	[(lines(1),6400), (largeGWs(8),27)]

18.7.1 Virtual Media Gateway (VMG) Names

The UE9000 MG is comprised of one or more VMGs. This section documents the format for naming of a VMG.

- **Format:** SSSSXXX-Y-Z
- **Where:**
- SSSS is a 1 to 4 character SITE name (as provisioned in Table SITE on the XA-Core).
- XXX is the office frame number (000-511).
- Y is the logical frame (0-7 for VMG created prior to SN07, fixed value of zero for VMG created in SN07 and later).
- Z is the shelf (0-3).

Example: LAKE017-0-2

18.7.2 Media Gateway Name

up to 32 characters with the following format:

<string>/<frame number>/<shelf number> where

<string> is MG9K_<n> where n is a number between 1 and 999 indicating the number of the MG9000 network element in this call server network. **<string>** cannot contain a “/”, but can contain alphanumerical and “_”.

<frame> is a number: 0-7

<shelf> is a number: 0-3

18.7.3 Line Termination Endpoint Names

tp/<slot>/<circuit> where

<slot> is a 2 digit number: 02-21

<circuit> is a 2 digit number: 00-31

The **tp** must be lower case.

Example: tp/03/01

18.8 CICM or 3rd party Large Line Gateways

The characteristics of the CICM and AFC gateways are listed in the following table.

Table 37: Gateway Profile Values and Configuration Parameters

Gateway Profile Name	GW Category	Signal Protocol	Protocol Version	Protocol Port	Service Type	Port/EP Capacity	GWC Profile Number
CICM	Large	MEGACO	1.0	2944	Line, ITRANS	3069	57

AFC	Large	MEGACO	1.0	2944	Line, Trunk, Audio, APG, DQOS, ITRANS	1023	67

18.8.1 Media Gateway Name

3rd party large line GW and SN06 CICM gateway naming is a 1 to 32 character alpha-numeric string. In SN07 the names for CICM is changed as shown below. AFC and other 3rd party large lane GW remain unchanged.

SN07 CICM gateway naming convention:

<name>-<nnn> where

<name> is any alpha numeric string

<nnn> is a zero padded number that ranges from 000 to 511. This number matches the frame number seen in the assigned logical group.

18.8.2 AFC Endpoint Names

Endpoints for AFC gateways takes the format

tp/<nnnn> where

<nnnn> is a zero padded number that range from 0000 to 1022.

18.8.3 CICM Endpoint Names

Endpoints for CICM gateways takes the format

Format of terminations provisioned in SN06:

tp/<nnnn>

<nnnn> is a number that ranges from 0 to 9999999. It is not zero padded.

Format of terminations provisioned SN07 and later:

tp/<G>/<nnnn> where

<G> is a number that ranges from 0-2.

<nnnn> is a number that ranges from 0000 to 1022. It is zero padded.

18.9 MTA Line Gateways

Capacities of MGs are defined in the profiles available for selection when the MG is associated to the GWC. Therefore, selection of capacity is important to get the maximum utilization of your GWC.

For example, a line GWC supports a maximum of 6400 datafilled ports, however, MTA devices support varying numbers of endpoints. Initial selection of reserved capacity should reflect the maximum number of lines this customer will possible use. This will reserve capacity in the GWC and may therefore impact the number of GWCs that support a given office. This is required to avoid the need for “rehoming” MTAs when unused lines are put into service.

A future release will allow lowering the reserved capacity for existing VMGs and to allow the customer to specify reserved port capacity upon creating the association.

Table 38: MTA Profile Values and Configuration Parameters for Provisioning

Profile Name	Service Type	Reserved Port Capacity	GUI Protocol	OSSGATE Protocol	Protocol Version
MOTOROLAMTA_1	line, DQOS	1	NCS	1	1.0
MOTOROLAMTA_2	line, DQOS	2	NCS	1	1.0
MOTOROLAMTA_4	line, DQOS	4	NCS	1	1.0
TOUCHTONE_NN01_1	line, DQOS	1	NCS	1	1.0
TOUCHTONE_NN01_2	line, DQOS	2	NCS	1	1.0

Table 38: MTA Profile Values and Configuration Parameters for Provisioning

Profile Name	Service Type	Reserved Port Capacity	GUI Protocol	OSSGATE Protocol	Protocol Version
TOUCHTONE_NN01_3	line, DOQS	3	NCS	1	1.0
TOUCHTONE_NN01_4	line, DQOS	4	NCS	1	1.0
ARRIS_TOUCHTONE_NN01_4	line, DOQS	4	NCS	1	1.0
ARRIS_TOUCHTONE_NN02_4	line, DQOS	4	NCS	1	1.0

Table 39: Internally Used MTA Profile Values and Configuration Parameters

Profile Name	MG Category	GWC Connectivity Profile Number	Protocol	Protocol Version (internal storage)
MOTOROLAMTA_1	small	52	1	1.0
MOTOROLAMTA_2	small	52	1	1.0
MOTOROLAMTA_4	small	52	1	1.0
TOUCHTONE_NN01_1	small	45	1	1.0
TOUCHTONE_NN01_2	small	45	1	1.0
TOUCHTONE_NN01_3	small	45	1	1.0
TOUCHTONE_NN01_4	small	45	1	1.0
ARRIS_TOUCHTONE_NN01_4	small	45	1	1.0
ARRIS_TOUCHTONE_NN02_4	small	52	1	1.0

Table 40: GWC Profiles used by MTA Line Gateways

Profile Name	Tone Data	([Exec, Term]...)	([Type,Capacity],...)
SMALL_LINENA	NORTHAA	[('POTSEX','POTS'), ('KSETEX', 'KEYSET')]	[(lines(1),6400), (smallGWs(7),6400), (ipsec(15), 0), (kerberos(16),0), (dqos(6),20)]
SMALL_LINEINTL	UKADSI	[('UTR250','PRAB'), ('DTCEX', 'ABTRK')]	[(lineports(1),6400), (smalllinegws(7),6400), (ipsec(15), 0), (kerberos(16),0), (dqos(6),20)]

18.9.1 Media Gateway Name

A domain name of the Media Gateway in the form of an absolute domain name including the hostname of the device and suitable for lookup using Directory Name Service (DNS). The name must contain a “.” and be no longer than 32 characters. An example would be:

cust34671.rdu.attcable.net

Steps to configure Cable gateway names where FQDN exceed 32 characters:

[TBD]

18.9.2 Line Termination Endpoint Names

aaln/<n> where

<n> is a number: 1-4 depending on the MTA model and market*

18.10 IAD Line Gateways

Capacities of MGs are defined in the profiles available for selection when the MG is associated to the GWC. Therefore, selection of capacity is important to get the maximum utilization of your GWC.

For example, a line GWC supports a maximum of 6400 datafilled ports, however, Integrated Access Devices (IADs) support varying

numbers of endpoints. Initial selection of reserved capacity should reflect the maximum number of lines this customer will possible use. This will reserve capacity in the GWC and may therefore impact the number of GWCs that support a given office. This is required to avoid the need for “rehoming” IADs when unused lines are put into service.

A future release will allow lowering the reserved capacity for existing VMGs and to allow the customer to specify reserved port capacity upon creating the association.

Table 41: IAD Profile Values and Configuration Parameters for Provisioning

Profile Name	Service Type	Reserved Port Capacity	GUI Protocol	OSSGATE Protocol	GUI Protocol Version	OSSGATE Protocol Version
MEDIATRIX_LINE_GW_4	line	4	MGCP	5	1.0	1.0
MEDIATRIX_LINE_GW_24	line	24	MGCP	5	1.0	1.0
ASKEY_LINE_GW_4	line	4	MGCP	5	1.0	1.0
ASKEY_LINE_GW_12	line	12	MGCP	5	1.0	1.0
ASKEY_LINE_GW_30	line	30	MGCP	5	1.0	1.0

Table 42: Internally Used MG Profile Values and Configuration Parameters

Profile Name	MG Category	GWC Connectivity Profile Number	Protocol	Protocol Version (internal storage)
MEDIATRIX_LINE_GW_4	small	49	5	1.0
MEDIATRIX_LINE_GW_24	small	49	5	1.0
ASKEY_LINE_GW_4	small	49	5	1.0
ASKEY_LINE_GW_12	small	49	5	1.0
ASKEY_LINE_GW_30	small	49	5	1.0
AMBIT_LINE_GW_16	small	49	5	1.0
MGCP_LINE_GW_1	small	49	5	1.0

Table 43: GWC Profiles used by IAD Line Gateways

Profile Name	Tone Data	[[Exec, Term]...]	[[Type,Capacity],...]
SMALL_LINENA	NORTHAA	[('POTSEX','POTS'), ('KSETEX', 'KEYSET')]	[(lines),6400), (smallGWs(7),6400), (ipsec(15), 0), (kerberos(16),0), (dqos(6),20)]
SMALL_LINEINTL	UKADSI	[('UTR250','PRAB'), ('DTCEX', 'ABTRK')]	[(lines(1),6400), (smallGWs(7),6400), (ipsec(15), 0), (kerberos(16),0), (dqos(6),20)]

18.10.1 Media Gateway Name

A domain name of the Media Gateway in the form of an absolute domain name including the hostname of the device and suitable for lookup using Directory Name Service (DNS). The name must contain a “.” and be no longer than 32 characters. An example would be:

cust34671.rdu.vendor.net

18.10.2 Line Termination Endpoint Names

aaln/<n> where

<n> is a number: 1-4 depending on the model and market*

Note: * size of media gateway is specified when choosing the media gateway profile. Size values reserve space on selected GWCs.

18.11 Other Gateway Controller Profiles

18.11.1 VRDN

The following profiles are supported for VRDN Gateway Controllers.

Table 44: VRDN GWC Profiles

Profile Name	Tone Data	[[Exec, Term]...]	[[Type,Capacity],...]
VRDN	NORTHAM	[('DTCEX', 'ABTRK')]	[(vrdn(12),0)]
VRDNINTL	UKADSI	[('DTCEX', 'ABTRK')]	[(vrdn(12),0)]

18.11.2 SIP-T

The following profiles are supported for SIP-T Gateway Controllers.

Table 45: SIP-T GWC Profiles

Profile Name	Tone Data	[[Exec, Term]...]	[[Type,Capacity],...]
SIP-T	NORTHAM	[['DTCEX', 'ABTRK']]	[(dpt(5),4096), (sipt(11),0)]
SIP-TINTL	UKADSI	[['DTCEX', 'ABTRK']]	[(dpt(5),4096), (sipt(11),0)]
SIP-T_APG	NORTHAM	[['DTCEX', 'ABTRK']]	[(apg(4), 2016), (dpt(5),4096), (apgGWs(10),1), (sipt(11),0)]
SIP-T_APGINTL	UKADSI	[['DTCEX', 'ABTRK']]	[(apg(4), 2016),(dpt(5),4096), (apgGWs(10),1), (sipt(11),0)]
SIP-T_APG_RA	NORTHAM	[['DTCEX', 'ABTRK']]	[(apg(4), 2016), (dpt(5),4096), (apgGWs(10),1), (sipt(11),0), (ra(13), 0)]
SIP-T_APG_RAIN TL	UKADSI	[['DTCEX', 'ABTRK']]	[(apg(4), 2016), (dpt(5),4096), (apgGWs(10),1), (sipt(11),0), (ra(13), 0)]

