VoiceGenie 7.2

OA&M Framework – CLC

User's Guide

About Genesys

Genesys Telecommunications Laboratories, Inc., a subsidiary of Alcatel-Lucent, is 100% focused on software for call centers. Genesys recognizes that better interactions drive better business and build company reputations. Customer service solutions from Genesys deliver on this promise for Global 2000 enterprises, government organizations, and telecommunications service providers across 80 countries, directing more than 100 million customer interactions every day. Sophisticated routing and reporting across voice, e-mail, and Web channels ensure that customers are quickly connected to the best available resource—the first time. Genesys offers solutions for customer service, help desks, order desks, collections, outbound telesales and service, and workforce management. Visit www.genesyslab.com for more information.

Each product has its own documentation for online viewing at the Genesys Technical Support website or on the Documentation Library CD, which is available from Genesys upon request. For more information, contact your sales representative.

Notice

Although reasonable effort is made to ensure that the information in this document is complete and accurate at the time of release, Genesys Telecommunications Laboratories, Inc., cannot assume responsibility for any existing errors. Changes and/or corrections to the information contained in this document may be incorporated in future versions.

Your Responsibility for Your System's Security

You are responsible for the security of your system. Product administration to prevent unauthorized use is your responsibility. Your system administrator should read all documents provided with this product to fully understand the features available that reduce your risk of incurring charges for unlicensed use of Genesys products.

Trademarks

Genesys, the Genesys logo, and T-Server are registered trademarks of Genesys Telecommunications Laboratories, Inc. All other trademarks and trade names referred to in this document are the property of other companies. The Crystal monospace font is used by permission of Software Renovation Corporation, www.SoftwareRenovation.com.

Technical Support from VARs

If you have purchased support from a value-added reseller (VAR), please contact the VAR for technical support.

Technical Support from Genesys

If you have purchased support directly from Genesys, please contact Genesys Technical Support at the following regional numbers:

Region	Telephone	E-Mail
North and Latin America	+888-369-5555 or +506-674-6767	support@genesyslab.com
Europe, Middle East, and Africa	+44-(0)-118-974-7002	support@genesyslab.co.uk
Asia Pacific	+61-7-3368-6868	support@genesyslab.com.au
Japan	+81-3-5649-6871	support@genesyslab.co.jp

Prior to contacting technical support, please refer to the Genesys Technical Support Guide for complete contact information and procedures.

Ordering and Licensing Information

Complete information on ordering and licensing Genesys products can be found in the Genesys 7 Licensing Guide.

Released by

Genesys Telecommunications Laboratories, Inc. www.genesyslab.com

Document Version: 09-2007 (2.9)

Table of Contents

Chapter

1 Introduction

The Command Line Console (CLC) provides a command line interface to many of the functionalities provided by the OA&M Framework. To access this information, users can issue commands through a telnet session to the CLC or via shell commands.

Chapter

2 Configuration

The Command Line Console configuration file is located at `/usr/local/cmp-proxy/config/cmpclc.cfg`. This file can't be modified manually if it is synchronized. The changes should be made via the CLC or the SMC. The parameter `cmp.clc_port` determines the port that can be used to access the CLC. By default this value is set to `8999`. Also, the parameter `cmp.externally_accessible_ips` can be used to allow access to the CLC. By default, access to the CLC is limited to localhost. Adding a hostname or IP to the `cmp.externally_accessible_ips` parameter will allow any connection requests from that hostname or IP to be accepted. Multiple hostnames/IPs can be specified by providing a pipe (|) delimited list of hostnames or IPs. For example:

```
cmp.externally_accessible_ips =
fibula.voicegenie.com|cmpdev.voicegenie.com
```

Chapter

3 Accessing the CLC

To access the CLC, users can telnet to the port defined by `cmp.clc_port`, which is by default port `8999`. Also, under Linux users can simply type `clc` at the command prompt of any VoiceGenie platform. On Windows it is preferable to use a telnet client like Putty to access the CLC via telnet.

On Linux a script called `clccmd` is included to allow scripts to be written that use the CLC to submit commands to the OA&M Framework. Script clccmd is located at `/usr/local/cmp-proxy/bin/clccmd`.

Notes: The telnet client that comes with the Windows 2000 Server operating system does not automatically echo back commands sent to the CLC. It is preferable to use another telnet client to access the CLC such as Putty. However, user can set local echo on by performing the following:

Access the CLC using telnet:

```
telnet localhost 8999
```

Enter the escape characters:

```
CTRL + ]
```

Set the telnet client to echo:

```
set LOCAL_ECHO
```

Pressing "`CTRL + C`" when using CLC from a Linux telnet client prevents further commands from being entered by the user. If the user accidentally enters "`CTRL + C`", they should enter the escape sequence (`CTRL + ]`) and type in "`quit`" to end the telnet session. Subsequently, the user can access the CLC again using the procedure as outlined above.

Chapter

4 Commands Overview

The CLC supports the following commands, where parameters in [] are required and those in <> are optional:

General Commands:

- query <filter string>
- health <service> <host> <instance>
- snapshot [service] <host> <instance>
- alarm [priority] [number] [data]
- runscript [host] [instance] [wait] [scriptlabel] <arguments>
- help <category>, ? <category>
- exit, x

General Component Operation Commands:

- tracelevel [service] <host> <instance> <lev>
- start <service> <host> <instance>
- stop <service> <host> <instance>

General Configuration Commands:

- cfginfo <service> <version>
- cfggetid [service] <host> <instance>
- cfgxmladd [service] “[subtype]” [version] [filename]

Configuration Editing Commands:

- cfgcreate [service] “[subtype]” [version] [filename] [configName]
- cfgupdate [cfgID] [filename]
- cfgdelete [cfgID]
- cfgtarget [cfgID] [service] <host> <instance>

Configuration Parameter Editing Commands:

- `paramquery [cfgID] <name>`
- `paramupdate [cfgID] [name] [value]`
- `paramenable [cfgID] [name]`
- `paramdisable [cfgID] [name]`

Provision Editing Commands:

- `prvquery <type> <service> <host> <instance>`
- `prvadd [type] [entry]`
- `prvupdate [entryID] [entry]`
- `prvdelete [entryID]`
- `prvtarget [entryID] [service] <host> <instance>`
- `prvuntarget [entryID] [service] <host> <instance>`

OA&M Network Editing Commands:

- `compadd [service] <host> <instance>`
- `compupdate [networkID] [host]`
- `compdelete [networkID]`

Other Generic Commands:

- `clearstats [service] <host> <instance> <args>`
- `clienttrace [service] <host> <instance> <args>`
- `login [service] <host> <instance> <args>`
- `logout [service] <host> <instance> <args>`
- `makeready [service] <host> <instance> <ags>`
- `restart [service] <host> <instance> <args>`
- `resume [service] <host> <instance> <args>`
- `rollover [service] <host> <instance> <args>`
- `sendevent [service] <host> <instance> <args>`
- `setstate [service] <host> <instance> <args>`
- `shutdown [service] <host> <instance> <args>`
- `suspend [service] <host> <instance> <args>`

Note: The CLC expects the result of a command within 4 seconds. If the result of the command is not returned to the CLC within 4 seconds the CLC with print out No response.

Chapter

5 General Commands

5.1 Query

The query command allows users to see what platforms and components are connected to the OA&M network. This command has the following format:

`query <filter string>`

where `<filter string>` is an optional argument that filters the results and shows only those lines that contain the specified string.

Example: `query galahad` shows all components with *galahad* in its hostname or component name; `query Call Manager` or `query callmgr` would show all Call Managers on the network; `query offline` would show all offline components.

Note: The CLC supports the use of short command names that are non-ambiguous, as a result, the command query can be shortened to `q`.

5.2 Health

The `health` command allows users to get the latest health string for a particular component on a server. The command has the following format:

`health <service> <host> <instance>`

where all parameters are optional. If no parameters are specified, the health for all components on the local system will be returned. `<service>` is the name of the service or component whose health information is being requested, i.e. `callmgr`, `clc`; the `<host>` is the hostname of where the service is located; and `<instance>` is the instance of the service, by default this is `1`. The format of the health status string returned is dependent on the specific component.

Example: `health` shows the health for all components on the local machine, `health callmgr` shows the health for the Call Manager on the local machine,

`health callmgr galahad` shows the health of the Call Manager on *galahad*. Typing `health clc` will show CLC health information which provides the following information:
`Health for Command Line Cnsl (clc) on <ip of CLC machine>`
`Started: <data/time when CLC process started>`
`Status: <ONLINE or OFFLINE>`
`Clients Connected: Current <currently connected CLC sessions>, Total <total CLC sessions connected since startup> Total Commands Issued: <total commands issued since CLC startup>`

5.3 Snapshot

The `snapshot` command gets a snapshot of the real time port status of a service. The command has the following format:

```
snapshot [service] <host> <instance>
```

where `[service]` is a service, usually Call Manager or VoiceXML Interpreter, `<host>` is an optional parameter that specifies the hostname and `<instance>` specifies the instance. If `<host>` is not specified a snapshot of the local `<service>` is returned.

5.4 Alarm

Using the CLC it is possible to inject alarms into the OA&M infrastructure. This is useful in writing monitoring scripts that need to send an alarm that is handled by the framework. The OA&M framework can route these alarms to syslog, SNMP or to a number of other sinks. This command has the following format:

```
alarm [priority] [number] [data]
```

where parameters in `[]` are required. The parameters are defined as follows:

- `[priority]` – a number between `0` and `5`, where `0` is critical (highest priority) and `5` is the lowest priority.
- `[number]` – a user defined number from `0` to `1048575` that can be used to uniquely identify the alarm.
- `[data]` – any user defined data that describes the alarm

This command will return `Alarm Sent` when it gets a reply that the log was sent correctly.

5.5 Run Script

The `runscript` command can be used to run scripts on a remote server. The scripts that can be run are defined in the CMP Proxy configuration using the

configuration parameter `cmp.script_labels`. Only scripts that are defined in the `cmp.script_labels` parameter can be run. This command has the following format:

```
runscript [host] [instance] [wait] [scriptlabel]
<arguments>
```

where parameters in `[]` are required and parameters in `<>` are optional. The parmeters are defined as follows:

- `[host]` – the name of the host that the command is directed towards, a `-` can be used to denote localhost
- `[instance]` – the instance of the service that the command is directed towards, a `-` can be used to denote 1
- `[wait]` – determines if the confirmation message sent back to the CLC should wait for the result of the script; valid values are `true` (i.e. wait for result) and `false` (i.e. do not wait for result)
- `[scriptlabel]` – the script label as defined in the CMP Proxy configuration parameter `cmp.script_labels`
- `<arguments>` – arguments for the script if any

Example: The following runs the script `viewlicense` (`/usr/local/cmp-proxy/scripts/viewlicense.pl`) and waits to return the result:

```
CLC> runscript 10.0.0.72 - true viewlicense get
Script /usr/local/cmp-proxy/scripts/viewlicense.pl get
succeeded, result is
vggateway in 2037/12/31 500
vggateway out 2037/12/31 500
vggateway asr 2037/12/31 500
vggateway tts 2037/12/31 500
```

5.6 Help

The `help` (or `?`) command returns general usage information for the CLC. The command is of the following format:

```
help <category>
```

where `<category>` is one of `config`, `provision`, `network` or `generic`. If `<category>` is not specified the general help information is returned, otherwise details for that specific category are returned.

5.7 Exit

The `exit` (or `x`) command can be used to gracefully close the connection to the CLC.

Chapter

6 General Component Operation Commands

6.1 Tracelevel

The `tracelevel` command allows users to get or dynamically set the `cmp.trace_flag` parameter of a service. This flag is responsible for turning on the detailed tracing information that is necessary for debugging. The command has the following format:

`tracelevel [service] <host> <instance> <lev>`

where the parameters are defined as follows:

- `[service]` – the name of the service that the command is directed towards
- `<host>` – the name of the host that the command is directed towards, a `-` can be used to denote localhost
- `<instance>` – the instance of the service that the command is directed towards, a `-` can be used to denote 1
- `<lev>` – if this parameter is not specified, the command returns the existing `cmp.trace_flag` state (`enabled`/`disabled`) for a service, other wise it can be set to `enable` or `disable` to enable or disable tracing.

Example: `tracelevel callmgr` returns the trace level of the local callmgr, whereas `tracelevel callmgr - - enable enable`s the local callmgr's trace

Note: Any changes in tracing made by the `tracelevel` command, i.e. enable or disable, are not permanent and will be reset to the configuration value of `cmp.trace_flag` once the service is restarted.

6.2 Start / Stop

The CMP Proxy is responsible for starting and stopping VoiceGenie software. The `start` and `stop` commands provide users with the ability to start or stop the platform or any individual component. The start and stop commands have the following format:

```
stop/start <service> <host> <instance>
```

The parameters are defined as follows:

- `<service>` – the name of the service that the command is directed towards, if this parameters is not specified it defaults to `all`, which specifies the entire platform
- `<host>` – the name of the host that the command is directed towards, a `-` can be used to denote localhost
- `<instance>` – the instance of the service that the command is directed towards, a `-` can be used to denote 1

After starting a component, the `query` command should be used until the component shows up as being online.

Notes: Only components that are defined in `cmp.components` can be started or stopped using the CLC.

VoiceGenie software can take up to 1 minute to stop. As a result, please be sure to wait at least 1 minute before trying to restart the software.

6.3 Resume / Suspend

The `resume` and `suspend` commands have the following format:

```
resume/suspend [service] [host] [instance] <force drop>
```

where parameters in `[]` are required and parameters in `<>` are optional. The parameters are defined as follows:

- `[service]` – the service that will be suspended or resumed
- `[host]` – the host that he service resides on, `-` can be used to denote localhost
- `[instance]` – the instance of the service, the default value is `1`
- `<force drop>` – this parameter can be `true` or `false`, determines if all existing processing should be stopped forcefully

Once the command is issued, CLC will display the command's result. The following table lists the possible command results:

Result	Meaning
Command is in processing	Command is valid and accepted by the platform.
Usage: … …	The command entered was of the wrong format.
Failed to process the command	The system can not process the command.

Note: The `resume` command is an alias for `setstate duplex;` the `suspend` command is an alias for `setstate disable.`

Chapter

7 General Configuration Commands

7.1 Cfginfo

The `cfginfo` command returns information to the user about what configurations exists for a given service and version. The command has the following format:

`cfginfo <service> <version>`

where parameters in <> are optional. The parameters are defined as follows:

- `<service>` – the service type of the configuration you are interested in
- `<version>` – the version of configuration you are querying for, version is specified in the format *x.y.z*

Note: If neither `<service>` nor `<version>` is specified, information about all configurations is returned by the command.

Example: The following is an example using the `cfginfo` command; it returns a list of all configurations since service and version are not specified:

```
CLC> cfginfo
CfgID   Component Type                  Version  Subtype        Name
-------------------------------------------------------------------------
   ----
2       CMP Server (cmpserver)           7.2.0  AS3.0          default
1       CMP Proxy (cmpproxy)             7.2.0  AS3.0          default-1
5       CMP Proxy (cmpproxy)             7.2.0  AS3.0          default-2
6       VG SNMP Agent (vgsnmp)           7.2.0  AS3.0          default
7       Call Manager (callmgr)           7.2.0  AS3.0          default
11      Call Manager (callmgr)           7.2.0  AS3.0          SIP
8       VXML Interpreter (vxmli)         7.2.0  AS3.0          default
12      VXML Interpreter (vxmli)         7.2.0  AS3.0          SIP
```

```
3       System Mgmt Cnsl (smc)          7.2.0  AS3.0          default
4       Command Line Cnsl (clc)         7.2.0  AS3.0          default
15      Command Line Cnsl (clc)         7.2.0  AS3.0          new config
9       Web Proxy (iproxy)              7.2.0  AS3.0          default
13      Web Proxy (iproxy)              7.2.0  AS3.0          SIP
10      Squid Cache (squid)             7.2.0  AS3.0          default
14      Squid Cache (squid)             7.2.0  AS3.0          SIP
```

Example: The following is an example using the `cfginfo` command; it returns a list of all configurations for the Call Manager since service is specified:

```
CLC> cfginfo callmgr
CfgID   Component Type                  Version  Subtype        Name
-------------------------------------------------------------------------
   ----
7       Call Manager (callmgr)            7.2.0  AS3.0          default
11      Call Manager (callmgr)            7.2.0  AS3.0          SIP
```

7.2 Cfggetid

The `cfggetid` command can be used to find out what configuration is being used by a service. The command has the following format:

`cfggetid [service] <host> <instance>`

where parameters in `[]` are required and parameters in `<>` are optional. The parameters are defined as follows:

- `[service]` – the name of the service for which the configuration ID is being queried
- `<host>` – the name of the host for where the service whose configuration ID is being queried, a `-` can be used to denote localhost
- `<instance>` – the instance number of the service whose configuration ID is being queried, default value is `1`, a `-` can be used to denote 1

Example: The following is an example of the `cfggetid` command. This example returns the configuration ID, configuration name and version of the configuration that is being used by the local Call Manager service:

```
CLC> cfggetid callmgr
Configuration ID: 11, Name: SIP, Version: 7.2.0
```

7.3 Cfgxmladd

The `cfgxmladd` command can be used to upload a new configuration XML for a service. This command should not be used under normal circumstances. The command has the following format:

`cfgxmladd [service] "[subtype]" [version] [filename]`

where parameters in `[]` are required. The parameters are defined as follows:

- `[service]` – the name of the service for which the XML is being added
- `[subtype]` – the subtype as specified in the XML file
- `[version]` – the version of the XML being added, the version is specified in the format *x.y.z*
- `[filename]` – the full path and filename of the XML file that is being added

Note: When an XML that already exists and is replaced, the CMP Server should be restarted so that changes in the XML take effect.

Chapter

8 Configuration Editing Commands

8.1 Cfgcreate

The `cfgcreate` command can be used to create a new configuration. The command has the following format:

```
cfgcreate [service] "[subtype]" [version] [filename] [configName]
```

where parameters in `[]` are required. The parameters are defined as follows:

- `[service]` – the type of service for which the configuration is being created
- `[subtype]` – the subtype of the configuration to be created
- `[version]` – the version of the configuration to be created, the version is specified in the format *x.y.z*
- `[filename]` – the full path and filename of the file to use to update values of the configuration, for the default values the user can specify -
- `[configName]` – the name of the new configuration being created

Example: The following is an example of the `cfgcreate` command. This example creates a configuration called "New Configuration" with all default values; it returns the configuration ID of the newly created configuration:

```
CLC> cfgcreate cmpproxy "AS3.0" 7.2.0 - New Configuration
Config ID: 16
cmp.log_dll.file: Parameter Saved
cmp.log_dll.upstream: Parameter Saved
cmp.log_dll.metrics: Parameter Saved
```

8.2 Cfgupdate

The `cfgupdate` command can be used to update an existing configuration with new parameter values. The command has the following format:

```
cfgupdate [cfgID] [filename]
```

where parameters in `[]` are required. The parameters are defined as follows:

- `[cfgID]` – the configuration ID of the configuration that should be updated
- `[filename]` – the full path and filename of the file to use to update values of the configuration

Example: The following is an example of the `cfgupdate` command. This example updates the configuration with ID 16 with values in the configuration file `/usr/local/cmp-proxy/config/cmpproxy.cfg`; it returns a list of parameters that were updated:

```
CLC> cfgupdate 16 /usr/local/cmp-
proxy/config/cmpproxy.cfg
cmp.backup_cmpe: Parameter Updated
cmp.components: Parameter Updated
cmp.primary_cmpe: Parameter Updated
pmli: Parameter Updated
cmgr: Parameter Updated
prxy: Parameter Updated
pmli_limits: Parameter Updated
cmgr_limits: Parameter Updated
prxy_limits: Parameter Updated
pmli_start_script: Parameter Updated
cmgr_start_script: Parameter Updated
prxy_start_script: Parameter Updated
pmli_restart_script: Parameter Updated
cmgr_restart_script: Parameter Updated
prxy_restart_script: Parameter Updated
pmli_stop_script: Parameter Updated
cmgr_stop_script: Parameter Updated
prxy_stop_script: Parameter Updated
```

8.3 Cfgdelete

The `cfgdelete` command can be used to delete an existing configuration. The command has the following format:

```
cfgdelete [cfgID]
```

where parameters in `[]` are required. The parameters are defined as follows:

- `[cfgID]` – the configuration ID of the configuration to be deleted

Example: The following is an example of the `cfgdelete` command. This example deletes the configuration with ID 16, it returns a confirmation of the delete:

```
CLC> cfgdelete 16
Configuration Deleted
```

Note: Configurations currently targeted to a component can't be deleted until it is no longer in use.

8.4 Cfgtarget

The `cfgtarget` command allows users to target a configuration to a specific serivce. The command is of the following format:

```
cfgtarget [cfgID] [service] <host> <instance>
```

where parameters in `[]` are required and parameters in `<>` are optional. The parameters are defined as follows:

- `[cfgID]` – the configuration ID of the configuration to be targeted
- `[service]` – the name of the service that this configuration is being targeted towards
- `<host>` – the hostname of machine where the service resides, a `-` can be used to denote localhost
- `<instance>` – the instance of the service that the command is directed towards, a `-` can be used to denote 1

Example: The following is an example of the `cfgtarget` command. This example targets configuration with ID 16 to the CMP Proxy on the localhost:

```
CLC> cfgtarget 16 cmpproxy
Configuration Targeted Successfully
```

Chapter

9 Configuration Parameter Editing Commands

9.1 Paramquery

The `paramquery` command can be used to query the entire contents or a single parameter of an existing configuration. The command has the following format:

```
paramquery [cfgID] <name>
```

where parameters in `[]` are required and parameters in `<>` are optional. The parameters are defined as follows:

- `[cfgID]` – the configuration ID of the configuration to be queried
- `<name>` – the configuration parameter to query, if blank the entire contents are queried

Example: The following is an example of the `paramquery` command. This example queries the value of the `cmp.trace_flag` parameter from the configuration with ID 16, it returns the value (`FALSE`) and whether the parameter is `Enabled` or `Disabled`:

```
CLC> paramquery 16 cmp.trace_flag
cmp.trace_flag: FALSE: Enabled
```

9.2 Paramupdate

The `paramupdate` command can be used to update the value of a parameter in a configuration. The command has the following format:

```
paramupdate [cfgID] [name] [value]
```

where parameters in `[]` are required. The parameters are defined as follows:

- `[cfgID]` – the configuration ID of the configuration to be updated
- `[name]` – the configuration parameter to update

- [value] – the new value of the parameter

Example: The following is an example of the paramupdate command. This example updates the value of the cmp.trace_flag parameter with the value TRUE:

```
CLC> paramupdate 16 cmp.trace_flag TRUE
cmp.trace_flag Parameter Updated
```

9.3 Paramenable

The paramenable command can be used to enable a parameter in a configuration. The command has the following format:

```
paramenable [cfgID] [name]
```

where parameters in [] are required. The parameters are defined as follows:

- [cfgID] – the configuration ID of the configuration to be enabled
- [name] – the configuration parameter to enable

Example: The following is an example of the paramenable command. This example enables the cmp.email parameter:

```
CLC> paramenable 16 cmp.email
cmp.email Parameter Updated
```

9.4 Paramdisable

The paramdisable command can be used to disable a parameter in a configuration. The command has the following format:

```
paramdisable [cfgID] [name]
```

where parameters in [] are required. The parameters are defined as follows:

- [cfgID] – the configuration ID of the configuration to be disabled
- [name] – the configuration parameter to disable

Example: The following is an example of the paramdisable command. This example disables the cmp.trace_flag parameter:

```
CLC>  paramdisable 16 cmp.trace_flag
cmp.trace_flag Parameter Updated
```

Chapter

10 Provision Editing Commands

10.1 Prvquery

The `prvquery` command can be used to query information about provisioning. The command has the following format:

`prvquery <type> <service> <host> <instance>`

where parameters in <> are optional. The parameters are defined as follows:

- `<type>` – the type of provisioning, a number, if it is not defined the list of valid types is returned
- `<service>` – the name of the component that the command is directed towards
- `<host>` – the name of the host that the command is directed towards, a `-` can be used to denote localhost
- `<instance>` – the instance on the component that the command is directed to, default value is `1`

Example: The following is an example of the `prvquery` command with no arguments. This command returns a list of valid provision types that can be used for the `<type>` parameter:

```
CLC> prvquery
Entry Type  Description
-----------------------------------------------------------------------
         1  DNIS - URL Mapping
         2  Dialing Rules
         3  SNMP Trap Filter
         4  Hunt Groups
       100  ICM Platform Mapping
       101  ICM Service Mapping
       102  ICM Trunk Mapping
```

```
       103  SS7 DNIS - URL Mapping
       104  SS7 Circuit ID/Media Platform Mapping
```

The following is an example of the `prvquery` command where the `<type>` is specified. In this case a list of all provisioning entries of that type is returned:

```
CLC> prvquery 1
Entry ID  Entry Type  Entry
-------------------------------------------------------------------
       1           1  <key name="DNIS" value="XXXX"/>
<application module="VXML">
<param name="url"
value="file:///usr/local/phoneweb/samples/helloaudio.vxml"/>
<param name="default" value="defaults.vxml"/>
</application>
```

The result is a table that contains a listing on Entry ID, Entry Type and the actual contents of the entry.

If the `prvquery` command is used where the `<type>` is specified as well as a specific component (i.e. service, host, instance), then all entries of that `<type>` that are targeted to that component are returned in the table along with the Entry ID.

10.2 Prvadd

The `prvadd` command can be used to add provisioning entries. The command has the following format:

```
prvadd [type] [entry]
```

where parameters in `[]` are required. The parameters are defined as follows:

- `[type]` – the type of provisioning, a number
- `[entry]` – the contents of the provision record, a new line is specified using `\n`

Example: The following is an example using the `prvadd` command, in the example an entry with type `1` is added, also, `\n` is used to insert new line characters where required:

```
CLC> prvadd 1 <key name="DNIS" value="XXXX"/>\n<application
module="VXML">\n<param name="url"
value="file:///usr/local/phoneweb/samples/helloaudio.vxml"/>\n<param
name="default" value="defaults.vxml"/>\n</application>
Entry Added with Entry ID 3
```

The following table outlines the return values and their meanings:

Result	Meaning
Entry Added with Entry ID <X>	Success, entry added with ID <X>.
Invalid Entry Key for Provision Type	Failed, entry key inside entry is not valid.

Result	Meaning
Database Error: Unable to Add Provision Entry	Failed, encountered a database error.
Unable to Add Entry: Duplicate Not Allowed	Failed, entry being added would create a duplicate value which is not allowed.

Note: The `prvadd` command does not perform validation of the entries that are added. As a result, it is possible to create invalid entries via the CLC. Users must be careful when using the `prvadd` command for adding provision entries.

10.3 Prvupdate

The `prvupdate` command can be used to update a provisioning entry. The command has the following format:

```
prvupdate [entryID] [entry]
```

where parameters in `[]` are required. The parameters are defined as follows:

- `[entryID]` – the ID of the entry that needs to be updated
- `[entry]` – the new contents of the entry, i.e. the updated value, a new line is specified using `\n`

Example: The following is an example using the `prvupdate` command, in the example an entry with ID 3 is updated, also, `\n` is used to insert new line characters where desired:

```
CLC> prvupdate 3 key name="DNIS" value="XXXX"/>\n<application
module="VXML">\n<param name="url"
value="file:///usr/local/phoneweb/samples/helloaudio.vxml"/>\n<param
name="default" value="defaults.vxml"/>\n</application>
Entry Updated
```

The following table outlines the return values and their meanings:

Result	Meaning
Entry Updated	Success.
Database Error: Unable to Update Provision Entry	Failed, encountered a database error.
Invalid Entry Key for Provision Type	Failed, entry key inside the entry is not valid.

Result	Meaning
Unable to Update Entry: Entry Key <X> should not be Changed	Failed, the entry key of the entry can not be changed using the `prvupdate` command, a new one must be added using `prvadd`.
Unable to Update Entry: Entry ID is Invalid	Failed, The entry ID provided is invalid.

Note: The `prvupdate` command does not perform validation of the entries that are updated. As a result, it is possible to create invalid entries via the CLC. Users must be careful in using the `prvupdate` command in updating provision entries.

10.4 Prvdelete

The `prvdelete` command can be used to delete provision entries. The command has the following format:

```
prvdelete [entryID]
```

The parameter is defined as follows:

- `[entryID]` – the ID of the entry that needs to be deleted

Example: The following is an example using the `prvdelete` command:

```
CLC> prvdelete 3
Entry Deleted
```

The following table outlines the return values and their meanings:

Result	Meaning
Entry Deleted	Success.
Database Error: Unable to Update Provision Entry	Failed, encountered a database error.
Unable to Update Entry: Entry ID is Invalid	Failed, invalid entry ID.

10.5 Prvtarget

The `prvtarget` command can be used to assign a provision entry to a component. The command has the following format:

```
prvtarget [entryID] [service] <host> <instance>
```

where parameters in [] are required and parameters in <> are optional. The parameters are defined as follows:

- [entryID] – the ID of the entry that will be assigned
- [service] – the component that the entry will be assigned to
- <host> – the hostname where the component is, use - or localhost to specify the localhost
- <instance> – the instance on the component that the command is directed to, default value is 1

Example: The following is an example using the prvtarget command where entry ID 1 is targeted to the local Call Manager:

```
CLC> prvtarget 1 callmgr
Provision with Entry ID 1 targeted to Network ID 6
```

The following table outlines the return values and their meanings:

Result	Meaning
Provision with Entry ID <X> targeted to Network ID <Y>	Success, entry <X> targeted to component <Y>.
Database Error: Unable to Target Entry ID <X> to Network ID <Y>	Failed, encountered a database error.
Internal Error: Unable to Target Entry ID <X> to Network ID <Y>	Failed, an internal error occurred, report to Support.
Invalid Entry ID: <X>	Failed, the entry ID provided is invalid.
Entry ID <X> already targeted to Network ID <Y>	The entry ID is already targeted to that component.
Duplicate Entry Key: Network ID <Y> already has an entry with the same Entry Key as <X>	Failed, entry can not be targeted since the component already has an entry with that entry key targeted to it (i.e. same DNIS).

10.6 Prvuntarget

The prvuntarget command can be used to remove a provision entry's association with a component. The command has the following format:

```
prvuntarget [entryID] [service] <host> <instance>
```

where parameters in [] are required and parameters in <> are optional. The parameters are defined as follows:

- [entryID] – the ID of the entry that will be assigned
- [service] – the component that the entry will no longer be assigned to

- `<host>` – the hostname of the component, use `-` or `localhost` to specify localhost
- `<instance>` – the instance on the component that the command is directed to, default value is `1`

Example: The following is an example using the `prvuntarget` command to remove a provision, with entry ID 1, from the local Call Manager:

```
CLC>  prvuntarget 1 callmgr
Entry ID 1 no longer targeted to Network ID 6
```

The following table outlines the return values and their meanings:

Result	Meaning
Entry ID <X> no longer targeted to Network ID <Y>	Success.
Database Error: Unable to Delete Entry ID <X> with target of Network ID <Y>	Failed, encountered a database error.
Invalid Entry ID: <X>	Failed, the entry ID provided is invalid.

Chapter

11 OA&M Network Editing Commands

11.1 Compadd

The `compadd` command can be used to add components to the OA&M network. The command has the following format:

```
compadd [service] <host> <instance>
```

where parameters in `[]` are required and parameters in `<>` are optional. The parameters are defined as follows:

- `[service]` – the service name of the component being added
- `<host>` – the hostname of where the component is being added, can be left blank to specify localhost, or use `-` or `localhost`
- `<instance>` – the instance of the component being added, blank or `-` implies instance 1

Example: The following is an example using the `compadd` command; an ICM Call Control component is added:

```
CLC> compadd icm localhost
Component Added with NetworkID 23
```

The following table outlines the return values and their meanings:

Result	Meaning
Component Added with NetworkID <Y>	Success, added with network ID <Y>.
Database Error: Unable to Add Component	Failed, encountered a database error.
Unable to Add Component: Invalid Component Type	Failed, component type is invalid.

Result	Meaning
Unable to Add Component: Corrupted Results	Failed, internal error, report to Support.
Unable to Add Component	Failed.

11.2 Compupdate

The `compupdate` command can be used to update both the name and hostname/IP of all components on the specified server. The command has the following format:

```
compupdate [networkID] [host]
```

where parameters in `[]` are required and parameters in <> are optional. The parameters are defined as follows:

- `[networkID]` – the networkID of the component whose hostname is being changed, it must be a CMP Proxy component.
- `[host]` – the new hostname of the component

Example: The following is an example using the `compupdate` command, updating the hostname to `foo.voicegenie.com`:

```
CLC> compupdate 9 foo.voicegenie.com
Component 9 Updated with Host foo.voicegenie.com and Name
foo.voicegenie.com
```

The following table outlines the return values and their meanings:

Result	Meaning
Component <Y> Updated with Host <Z> and Name <Z>	Success, update <Y> with the hostname <Z> and also changed name to <Z>.
Unable to Update Component: Invalid NetworkID	Failed.
Database Error: Unable to Update Component	Failed, encountered database error.

11.3 Compdelete

The `compdelete` command can be used to delete a component. The command has the following format:

```
compdelete [networkID]
```

The parameter is required and is defined as follows:

- `[networkID]` – the networkID of the component which is being deleted, a cmp proxy component can only be deleted if it has no child components.

Example: The following is an example using the `compdelete` command, deleting networkID 23:

```
CLC> compdelete 23
Component with NetworkID 23 Deleted
```

The following table outlines the return values and their meanings:

Result	Meaning
Component with NetworkID <Y> Deleted	Success.
Unable to Delete Component: Invalid NetworkID	Failed.
Unable to Delete Component: Children must be Deleted First	Failed, can not delete a component that has children, i.e the CMP Proxy can not be deleted without deleting all other components on that server first.
Unable to Delete Component: Component must be offline	Failed, component must be offline in order to delete it.
Database Error: Unable to Delete Component	Failed, encountered database error.

Chapter

12 Other Commands

12.1 Rollover

The `rollover` command can be used to rollover the `pw_metricsfile` if required. The command has the following format:

`rollover cmpproxy <host> <instance>`

where parameters in <> are optional. The parameters are defined as follows:

- `<host>` – the hostname of the server where the rollover should occur, can be left blank to specify localhost, or use `-` or `localhost`
- `<instance>` – the instance, blank or `-` implies instance 1

Example: The following is an example using the `rollover` command:

```
CLC> rollover cmpproxy
Metrics file rolled over
```

12.2 Sendevent

The `sendevent` command can be used to send messages to an active VoiceXML session. The command has the following format:

`sendevent vxmli [host] [instance] [recipient_session_id] [sender_address] [message]`

where all parameters are required. The parameters are defined as follows:

- `[host]` – the host where the VXMLI is running. For localhost, a `-` can be used
- `[instance]` – the instance of the interpreter, `-` implies instance 1
- `[recipient_session_id]` – the unique session ID of the VXMLI session the message will be sent to. A typical session ID appears as follows:

  `0C9703E2-0C0028ED-0001`

The session ID of a VoiceXML session can be determined in different ways, including:

- The value of the `session.com.voicegenie.instance.myself` session variable of that session.
- The value of the X-Session-Id HTTP header in the HTTP requests performed by that session.

- `[sender_address]` – can also be specified as a valid session ID of a VXMLI session, or any character string whose length is under 127 characters. If the `sender_address` is a valid session ID, the recipient session would be able to send back a message to the session with the ID specified by the `sender_address`, which is dependent on the VoiceXML application.
- `[message]` – *any* combination of characters, with a maximum length of 2999 characters.

Example: The following is an example using the `sendevent` command:

```
CLC> sendevent vxmli - - 0C9703E2-0C0028ED-0001 1234
Hello World!
```

The following table outlines the return values and their meanings:

Result	Meaning
Sending Message from <sender_address> to <recipient_address>	Success.
Usage: sendevent [service] [hostname] [inatnce][recipient_address] [sender_address] [message]	Failed, invalid format command.
Can not deliver message	Failed.
Failed to send message	Failed.

Chapter

13 Call Manager Specific Commands

The CLC supports the following Call Manager specific command:

- `setstate`

13.1 Setstate

The `setstate` command has the following format:

```
setstate callmgr [Hostname] [Instance] [Mode] <Board>
<Channel> <Force Drop>
```

Note: Parameters in <> are optional, [] mandatory.

where:

- `[Hostname]` is the hostname of the machine where the command is targeted.
- `[Mode]` sets the administrative state to `inbound/outbound/duplex/disable/config.`
  - `inbound` – Set the given platform/board/channel to only allow inbound calls.
  - `outbound` – Set the given platform/board/channel to only allow outbound calls.
  - `duplex` – Set the given platform/board/channel to allow both inbound and outbound calls.
  - `disable` – Set the given platform/board/channel to not allow inbound nor outbound call.
  - `config` – Set the given platform/board/channel to re-read configuration from `~pw/glines.cfg` (for dialogic only).

Note: For SIP, only `outbound` and `duplex` states are supported. Also, for SIP, `inbound` state will work the same as `duplex`, while `disable` operates the same as `outbound`.

- `<Board>` is the optional board ID.

Note: For SIP, the board ID should be `0`. For Dialogic boards, the board ID should be the board index (starting from `1`) plus `100`. As a result, valid board IDs are from `101` through `108`. These IDs correspond to the board configuration in `~pw/glines.cfg`. A board ID of `-1` represents all boards and is the default value.

- `<Channel>` is the optional channel ID.

Note: For Dialogic, valid channel IDs are `1` through `30`. For SIP, channel ID is ignored. By default, the channel ID is `-1`, which represents all channels on the given board.

- `<Force Drop>` determines if the call should be dropped immediately. Valid values are `true` and `false`. If it is `true` and if required (say, switching from `inbound` to `outbound` mode), it will drop any existing calls immediately. If it is `false`, the operation will not be executed until the current calls have gracefully terminated. The default value is `false`.

Once the command is issued, CLC will display the command's result. The following table lists the possible command results:

Result	Meaning
Command is in processing	Command is valid and accepted by the platform.
Invalid board ID	The Board ID entered is invalid.
Given Board does not exist	The given board does not exist on the platform.
Invalid channel ID	The Channel ID entered by the user is invalid.
Usage: … …	The command entered was of the wrong format.
Failed to process the command	The system can not process the command.

If the board number is only provided, the operation will be carried out on all channels on the specified board. If neither board nor channel is provided, the operation will be carried out on all boards and channels on the platform.

Example: provided that hostname of the desired VG platform is `cmpdev:`

`setstate callmgr cmpdev - duplex 101 22` would trigger VG platform to set board 1 (dialogic) channel 22 to `duplex` mode, and `setstate callmgr`

`cmpdev duplex` would trigger VG platform to set all channels in `duplex` mode.

Note that the management of the states for platform, board and channels are all independent. The ultimate operational state of a channel is decided by the combination of the states of all three levels, that the most restricted state will apply. Here are a few examples:

Platform Admin State	Board Admin State	Channel Admin State	Channel Final Operational State
DISABLE	DUPLEX	DUPLEX	DISABLE
DUPLEX	DUPLEX	INBOUND	INBOUND
OUTBOUND	INBOUND	DUPLEX	DISABLE

Considering a VG platform with 2 boards, with all states set to `DUPLEX` on startup:

- user sets board 1 to `INBOUND` (i.e. `setstate callmgr [Hostname] inbound 101`)
- user sets platform to `DISABLE` (i.e. `setstate callmgr [Hostname] disable`)
- user sets platform to `DUPLEX` (i.e. `setstate callmgr [Hostname] duplex`)

In this scenario after the VoiceGenie server restarts, board 1 will be set to take on inbound calls.

Appendix

VoiceGenie SNMP Traps

This Appendix contains the list of traps that can be produced by the CLC component. The second last number in the OID corresponds to the component type. The component type for CLC is `308`.

The last number in the trap corresponds to the LogID, this ID uniquely identifies the log and corresponds to the ID field in the CallLog table.

Each `Description` field contains information about the relative severity of the alarm, the severities are:

Severity	Description
CRIT	An alarm event that denotes a critical or fatal condition and results in the failure of the software.
EROR	An alarm event that denotes an error condition that should never happen and that results in the loss of functionality.
WARN	An alarm event that denotes an exceptional situation that may occur legitimately but it is necessary to be aware of.

Also, each alarm has a Response Code specified; the response codes are defined as follows:

Severity	Description
CKAPP	Check application or Web server
CKCFG	Check and correct configuration
CKFS	Check file/directory existence/permission, or disk space
CKHW	Check hardware
CKTY	Check telephony hardware/connection
CKASR	Check ASR server
CKTTS	Check TTS server

Severity	Description
CKNW	Check network connection
CKOP	Check operational state of the server
NOTE	Notice/observation
REVG	Report to Support `[with logs]`
SWRS	Software restart: `[collect logs]` restart VoiceGenie server
HWRS	Hardware restart: `[collect logs]` reboot VoiceGenie platform

In addition the table lists any old alarm code (i.e. pre 6.4) that may relate to the alarm.

A.1 CLC Traps

The prefix for all CLC OIDs is `.1.3.6.1.4.1.7469.251.1.308.`

Name	Log ID	Level	Description	Causes	Response Code/Detailed Recommended Action
VGLOG-INVALID-MSG-TYPE-CMPC	1048597	WARN	Invalid Message Type Sent or Received	The CMP Agent received a message of unknown type	REVG – Report to Support, a highly unlikely event, could be caused by deployment with mixed versions.
VGLOG-CANNOT-CREATE-CLIENT-CMPC	1048599	WARN	Error Creating Client Socket	The CMP Agent is unable to create a client socket	REVG – Report to Support, a highly unlikely event, could be caused by machine wide socket issues.
VGLOG-DSRV-INVALID-QUERY-CMPC	6291477	WARN	Invalid data query string received	Data Service is requesting information for an unknown parameter	REVG – Could be a misconfiguration, report to Support.
VGLOG-DSRV-UNSUPPORTED-VAR-CMPC	6291478	WARN	Query for unsupported variable received	Data Service is requesting information for an unknown parameter	REVG – Could be a misconfiguration, report to Support.

Name	Log ID	Level	Description	Causes	Response Code/Detailed Recommended Action
VGLOG-CANNOT-CREATE-SERVER-CMPC	1048598	WARN	Error Creating Server Socket	The CLC creates a server socket for user connections, this can not be created.	REVG – Report to Support, a highly unlikely event, could be caused by machine wide socket issues.
VGLOG-CFG-WRITE-FAIL-CMPC	2097178	WARN	Configuration file could not be written	File may be read only or inaccessible.	CKCFG – Check that the configuration file is writeable by the phoneweb user.
VGLOG-VGASSERT-CMPC	135267305	CRIT	VGASSERT	This is a general assert.	REVG – The situation where this arises can be varied, must be reported to Support.
VGLOG-SOCKET-SEND-FAILED	134219731	EROR	Socket send failed	Unable to send data over a socket, could be a number of socket related problems.	CKNW/REVG – Report to Support, a highly unlikely event, could be caused by machine wide socket issues.
VGLOG-TEST-ALARM	1	WARN	CMP Test Alarm	Used to generate a test alarm from the CLC.	–
VGLOG-DB-REPAIR-ALARM	1000	WARN	CMP Database table needs repair	The database monitoring scripts have found that a table in the database needs to be repaired.	REVG – Report to Support.
VGLOG-DB-REPAIR-FAIL-ALARM	1001	EROR	Repairing CMP Database table failed	The database monitoring scripts have attempted a repair that has failed.	REVG – Report to Support.

Name	Log ID	Level	Description	Causes	Response Code/Detailed Recommended Action
VGLOG-DB-TRUNCATE-ALARM	1002	WARN	Truncating a CMP Database table	The database monitoring scripts have found that a large table in the database needs to be repaired and will be done so by truncating.	REVG – Report to Support.
VGLOG-DB-TRUNCATE-FAIL-ALARM	1003	EROR	Truncating of a CMP Database table failed	The database monitoring scripts have attempted a truncate that has failed.	REVG – Report to Support.
VGLOG-DB-QUEUE-EXEC-ERROR-ALARM	1004	WARN	There was an error obtaining the queue length from the CLC.	The database monitoring scripts could not access the CLC or could not parse the queue length information from the response given by the CLC.	REVG – Check that the CLC is operational and that the health information from the CMP Server (`CLC> health cmpserver`) is getable. Send output to Support.
VGLOG-DB-QUEUE-LENGTH-ALARM	1005	EROR	The queue has exceeded the maximum length allowed, %d.	This may be caused by the database (MySQL) being used by another application, causing the MySQL server to slow down and increasing the queue. Also, it could be the case that the CMP Server is being inundated with logs and can not handle the data throughput.	REVG – Log in to MySQL (`mysql -u pw -ppw NDM`) and get the out put of `show processlist;` to see what processes are using the database. Also, turn down any excessive logging going to the database (i.e. metrics, tracing, etc.)

Name	Log ID	Level	Description	Causes	Response Code/Detailed Recommended Action
VGLOG-DB-QUEUE-INCREASING-ALARM	1006	CRIT	The queue has increased in size for too many consecutive sample periods, %d.	This may be caused by the database (MySQL) being used by another application, causing the MySQL server to slow down and increasing the queue. Also, it could be the case that the CMP Server is being inundated with logs and can not handle the data throughput.	`REVG` – Log in to MySQL (`mysql -u pw -ppw NDM`) and get the out put of `show processlist;` to see what processes are using the database. Also, turn down any excessive logging going to the database (i.e. metrics, tracing, etc.)
VGLOG-DB-REPLICATION-EXEC-ERROR-ALARM	1007	WARN	There was an error in obtaining the replication status.	The database monitoring scripts could not access MySQL to get the status of the replication.	Check that the root password is added correctly to the configuration using the `dbadmin.jar addpass` command. Check that the MySQL client works and connects to the database.
VGLOG-DB-REPLICATION-BROKEN-ALARM	1008	CRIT	The replication has been broken and needs to be repaired.	The database slave is no longer running correctly on the database server and as a result database replication is broken.	Database replication must be set up again.

Name	Log ID	Level	Description	Causes	Response Code/Detailed Recommended Action
VGLOG-DB-BINLOG-EXEC-ERROR-ALARM	1009	EROR	There was an error in obtaining the bin log status.	The script could not access MySQL to get the status of the binary log files or could not purge the files, MySQL may not be running, or replication may not be setup correctly.	Restart MySQL or resetup replication.
VGLOG-DB-BINLOG-DELETE-ALARM	1010	NOTE	Binary logs have been deleted up to log %s.	Used to inform the customer that a purge has occurred.	– (Normal event)
VGLOG-CMP-DEPLOYER-FAILED-ALARM	2000	CRIT	The CMP Deployer has failed to start correctly, check that the CMP Server, CMP Proxy and CLC are functional	The CMP Server, CMP Proxy or CLC may not be functional.	Check the CMP Server, CMP Proxy and CLC configuration and ensure that they are started.

Revision History

Version	Date	Change Summary	Author/Editor
1.0	August 13th 2003	Initial release	Rakesh Tailor
2.0	September 19th 2003	Updated configuration files in appendices. Update sections 2.1.1, 5.1, 7. Added 6.4 for Provisioning service.	Monti Ghai
2.1	September 23rd 2003	Updated sections on SMC configuration as well as details on Hunt Groups and Dialing Rules.	Wen Wang
2.2	December 17th 2003	Updated document to reflect changes for CMP2.1.	Rakesh Tailor
2.3	March 2nd 2004	Updated document to reflect changes for CMP2.2	Rakesh Tailor
2.4	June 19th 2004	Added details for new features in CMP2.3, including Logging, Alarming and SNMP changes	Rakesh Tailor
2.5	February 28th 2005	Added details for new features in VoiceGenie 7.0.0	Rakesh Tailor
2.6	April 13th, 2005	Final Revision for VoiceGenie 7 Release	Andrew Ho
2.7	March 14th, 2006	Updates for VoiceGenie 7.1.	Rakesh Tailor
2.8	September 5th,2006	Updates for VoiceGenie 7.1	Monti Ghai
2.9	September 21st, 2007	Updates for VoiceGenie 7.2	Wen Wang